General Knowledge Representation And Sharing For Disaster Management

Philippe A. Martin [1][0000-0002-6793-8760] and Tullio J. Tanzi [2][0000-0001-5534-7712]

[1] EA2525 LIM, University of La Réunion, F-97490 Sainte Clotilde, France
philippe.martin@univ-reunion.fr
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, Paris, France
tullio.tanzi@telecom-paris.fr

Abstract. The first part of this article first distinguishes "restricted knowledge representation (KR) and sharing (KS)" and the still seldom researched task of "*general* KR and KS". This parts then highlights the usefulness of the latter for disaster management, and provides a panorama of complementary techniques supporting it. The research question that these techniques collectively answer is: how to let Web users collaboratively build KBs (KR bases) i) that are not *implicitly* "partially *redundant* or *inconsistent*" internally or with each other, ii) that are complete with respect to certain criteria or subjects, iii) without restricting what the users can enter nor forcing them to agree on terminology or beliefs, and iv) without requiring people to duplicate knowledge in various KBs, or manually search knowledge in various KBs and aggregate knowledge from various KBs? In a second part, this article shows the way various kinds of disaster management related information can be categorized or represented for general KS purposes, e.g. terminologies and information objects (these objects are rarely represented via KRs; examples about *Search & Rescue* procedures are given).

Keywords: Disaster Management, Knowledge Sharing, Ontology.

1 Introduction

Disaster management, e.g. disaster risk reduction or *Search & Rescue* operations, requires many resources, e.g. certain kinds of people, search robots, maps, communication tools, detection devices, search procedures and search software. It also depends on many parameters, e.g. the available resources, the nature of the disaster, the terrain and the weather. Ideally, all published information on all these elements would be stored, related and organized on the Web in places and in ways permitting people and software agents to i) retrieve and compare them according to any set of criteria, and ii) complement the stored information in ways that maintain its degree of organization and hence retrievability.

Such an ideal and scalable organization of information implies the building and exploitation of *knowledge representation* bases: *KR bases* or simply *KBs*. KBs store KRs (alias, *knowledge*), i.e. *semantic and logic-based* representations of information. In this article, KRs is opposed to *data*, i.e. information merely organized by predefined *structural* relations (i.e. partOf ones) and *semantic relations of very few predefined types* (mostly typeOf relations). In KBs, all the types (i.e. relation types and con-

cept types) and their definitions are user-provided: most of the knowledge in many KBs are expressed via the definitions. Document based technologies and database systems only handle *data*, although deductive databases are steps towards KBs. A KB is composed of an ontology and a base of facts. An ontology is i) the set of terms used in representations within KBs, along with ii) representations of term definitions, hence semantic relations between the terms. Natural language based documents or databases cannot automatically be converted into *KBs that are well-organized via generalization (and part-of) relations*, if only because they often lack the necessary information for even human readers to derive such relations. This organization – and hence, manually or semi-automatically built KBs – is necessary to support i) *semantic*-based searches, via queries or navigation, and ii) any *scalable* organization or integration of information. This is why methodologies or architectures for building ontologies or ontology based systems, and their advantages, have already often been discussed in relation to disaster management related information. E.g., in July 2020, the digital library of IFIP (International Federation for Information Processing) included 12 articles about ontologies and "risk or disaster or emergency", while the digital library of the ISCRAM conferences ("Information Systems for Crisis Response and Management" conferences) included 46 articles in which "ontology" was recorded as a keyword.

There are some *top-level* ontologies related to disaster management, e.g. SEMA4A [5] (the purpose of which is to help alerting people of imminent disasters) and *empathi* [3] which is more general and integrates some other top-level ontologies. However, as of 2020, there does not appear to be any publicly accessible large *content ontology* about information related to disaster management, let alone KBs in which people or organizations could relate or aggregate information. For instance, even though [14] mentions past "massive efforts e.g. in European projects such as DISASTER (disaster-fp7.eu), SecInCoRe (www.secincore.eu), EPISECC (www.episecc.eu), or CRISP (www.crisppproject.eu)", only the second and third project Web pages of this list are now accessible, and the results of these projects are not KBs but reports about planned works and advocated architectures or small models (top-level ontologies). Some other large projects such as the Norwegian INSITU (Sharing Incident and Threat Information for Common Situational Understanding) project (2019-2022) [13] focus more on terminology harmonisation as well as tools for the *collaborative synthesis* of information in classic media (textual documents, databases, maps, ...), hence not information synthesis via KBs. The use of classic media make terminology harmonisation useful for lexical searches but this harmonisation is a complex task requiring committees (hence centralization) and it is useful only when its guidelines are followed (which is not easy to do). With KBs, such terminology harmonisation is not necessary: relations of generalisation or equivalence between terms within KBs or across KBs can be added in a decentralized and incremental way by each term provider or knowledge provider. Thanks to these relations, people can use the terms they wish without decreasing knowledge retrievability.

There are two meanings for "knowledge sharing" (KS). "*Restricted KS*" is quite related to *data(base) sharing*: it is about i) easing the exchange of information (data or KRs) between *particular* agents (persons, businesses or applications) that *can* discuss

to solve ambiguities or other problems, and ii) the *full or efficient* exploitation of the information by these particular agents. "*General KS*" is about people representing or relating information within or between KBs in ways that maximize the retrievability and exploitation of the information by *any* person and application. These two meanings are very rarely explicitly distinguished, including by the World Wide Web Consortium (W3C). Regarding KS, the W3C has a "Semantic Web vision of a Web of *Linked data* [17]". As the name may suggest, and as explained in Section 2, the vision and techniques proposed by the W3C are mainly focused on *restricted KS*. Risk management related research articles that advocate the use of KBs, e.g. [2] and [4], rely on the W3C approach or techniques. However, they are insufficient for (general) KS in disaster management, if only given the amount of potentially useful information for such a management. This insufficiency is also one reason for the above cited lack of large publicly accessible content ontologies or KBs related to disaster management.

Section 2 summarizes complementary ways to support *general KS* and the above cited ideal. By doing so, Section 2 also explains different aspects of the above cited insufficiency. The originality of Section 2 is mainly in the synthesis or panorama itself, more than in the description depth of the cited or introduced techniques, because the first author has separately published on several of these techniques. However, new elements have been introduced and these techniques are original wrt. disaster management related articles. Furthermore, these techniques or ways to support general KS *collectively* answer the following research question: how to let Web users collaboratively build KBs i) that are not *implicitly* "partially *redundant* or *inconsistent*" internally or with each other, ii) that are complete with respect to certain criteria or subjects, iii) without restricting what the users can enter nor forcing them to agree on terminology or beliefs, and iv) without requiring people to duplicate knowledge in various KBs, or to manually search knowledge in various KBs and aggregate knowledge from various KBs?

Using various examples, Section 3, the second part of this article, shows the way various kinds of disaster management related information can be categorized or represented for general KS purposes. Section 3.1 illustrates the representation and organization of a small terminology, and the advantages of performing such tasks. Section 3.2 gives a general model to represent and organize *Search & Rescue* information; the illustrated originality is the handling of information objects. Section 3.3 gives KRs about automatic explorations of a disaster area, e.g. by a rover; the illustrated originality is the representation of procedures. http://www.webkb.org/kb/nit/o_risk/ gives access to the full representations that these sections illustrate, as well as other ones: representations synthesizing and organizing the important content of three related articles about how to create rovers adapted to a terrain, the biggest article being [15].

2 Complementary Ways to Support General Knowledge Sharing

2.1 Tools To Import & Export Any Kind Of Knowledge, Even In User Specified Formal Languages

Knowledge representations (KRs) are logic statements. For graph-oriented models, KRs are concept nodes (i.e. possibly quantified instances of some types) possibly connected by relation nodes (existentially quantified instances of relation types). The *non-predefined terms* in KRs are defined or declared in ontologies by knowledge providers and are identifiers of either individuals (type instances that cannot have instances) or types. Types are either relation types or concept types. Section 3 gives examples.

Regarding KR languages (KRLs), the W3C proposes i) some ontologies of logic models – e.g. RDF for the simplest formulas (the existential quantified conjunctive ones), OWL for the SROIQ description logic, RIF for rule-based more expressive classic logics – and ii) some notations for some models, e.g. the notations RDF/XML, RDF/Turtle and RIF/XML. There are other standards for KR logic models, e.g. Common Logic (CL, the ISO/IEC 1st-order logic model), with various notations for them, e.g. CL/XML (XCL). However, as described by the next two paragraphs, all these KRLs have problems for general KS, hence for general disaster management purposes.

The first problem of these KRLs is their expressiveness restrictions. Although these restrictions ensure that what is represented has particular interesting properties (e.g. efficiency properties), this is at the cost of *preventing* the representation of some information: some KRs are either not written, hence not shared, or written in ways that are *ad hoc*, biased or imprecise, hence in far less exploitable ways. For general KS, using expressive KRs has often no downside since, whichever their expressiveness, KRs can be translated into less expressive ones, often automatically, to fit the need of a particular application, by *discarding* the information that *this* application does not handle or require. On the other hand, KRs designed for particular applications are often unfit (too biased, ...) for other ones. Since present or future disaster management is not a fixed list of known applications, expressiveness restrictions limit it.

Another important problem of these KRLs is that they are not high-level in the sense that they do not allow "normalized and easy to read or write" representations for many useful notions such as meta-statements, numerical quantifiers and interpretations of relations from collections. Thus, even when representing similar information, different KRs written in different languages or by different users are difficult to translate and match automatically, and hence to search or aggregate. The use of *ontology design patterns* (e.g. those of [1]) by knowledge providers only very partially addresses these issues and is difficult, hence rarely performed. Furthermore, for different domains or applications, different notions and different ways to write them are useful. Creating or visualizing KRs via the current KR editors is even more restricting in terms of what can be expressed and displayed. E.g., graphics take a lot of space and hence do not allow people to simultaneously see and hence visually compare a lot of KRs (this is problematic for KR browsing and understanding).

One answer to these problems was i) FL [8], a very expressive, concise and configurable textual notation for KRs, and ii) FE [6], a more verbose version of FL which

looks like English and hence is more easily read by non-experts. A complementary and more general answer is the creation of an ontology of not only model components for logics but also of notation components for them. KRLO (KRL ontology) [10] is a core for such an ontology: it allows people to specify any KR language they wish. Software modules that exploit such an ontology are currently being designed. With these modules, KB systems will be able to import and export from, to or between any such specified KR languages, and hence also perform certain kinds of KR translations (furthermore, since the rules for these translations are also specified in the ontology, tool users may select the rules to apply and they may also complement these rules).

2.2 General-purpose Ontologies Merging Top-level Ontologies and Lexical Ones

Top-level ontologies define types that support and guide the representation, organization and checking of KBs. Lexical ontologies organize and partially define the meanings of common words and relate these meanings to these words. Both kinds (top-level ones and lexical ones) are domain-independent, hence reusable for disaster management. The more types from such ontologies a KB reuse, the easier it is to create and organize its content and the more this content can be retrieved via these types. The more types from such ontologies two independently created KBs share and are based on, the easier the content of these two KBs can be aligned or fully integrated. Since such ontologies are sets of definitions, not assertions of facts or beliefs, inconsistencies between them are signs of conceptual mistakes, e.g. misinterpretations or over-restrictions. Thus, when not fully redundant, such ontologies are complementary and, possibly after some corrections, may be merged without leading to inconsistencies.

The Multi-Source Ontology (MSO) [7] is one step towards such a merged ontology. It already merges many top-level ontologies and a lexical ontology derived from WordNet. More will be added. Unlike for other merges, Web users can cooperatively complement and improve the MSO via the methods described in the next subsection. In accordance with these methods, the top-level of the MSO has already been organized via subtype partitions, and hence has advantages similar to those of a decision tree for knowledge retrieval and inference purposes. Furthermore, the MSO includes KRLO and hence types that are interesting for representing or categorizing procedures or software. As illustrated in Section 3.2, this last point is useful too for disaster management purposes.

2.3 KB Servers That Support Non-restricting KB Sharing By Web Users

For general KS, a KB should not include two statements logically inconsistent with one another, since classic logics – and hence most inference engines – cannot handle KBs that are logically inconsistent. Yet, different users of a shared KB may want to enter statements that happen to be inconsistent with each other. For general KS, the avoidance of inconsistencies in a shared KB cannot be done by the owner(s) of each shared KB *accepting or not* each statement submitted to the KB: this not only defeats the purpose ("*general* KS"), this is also a too slow and arbitrary process to be scalable. Automatically dispatching submitted statements into various KBs for each one

to be internally consistent, is also not scalable: with such a method, the number of required KBs and redundancies between KBs can grow exponentially.

The solution starts by associating each term *and each statement* to its source (document or author). For terms, this is now standard practice: the W3C advocates the systematic use of URLs, with the possible use of abbreviations for the sources, but there are other more flexible and scalable solutions. For statements, making this association is recognizing that facts in KBs are actually *beliefs*; this may be formalized using meta-statements that contextualize statements according to who created them or believe in them. (Unfortunately, the W3C has not yet made recommendations regarding this last point and OWL does not handle meta-statements). In such KBs, statements may be seen as either "beliefs" or "definitions". Since these last ones are tied to a term, they cannot be false, i.e. they are true "by definitions": the meaning of the term is whatever its definitions specify. E.g., assuming that `pm` identifies a particular user in a KB, then `pm` has the right to create the term `pm:Table` (this term identifier uses the term prefixing syntax usable in most W3C KRLs) and define it as a type for flying objects rather than as a type of furniture. Thus, *definitions* need not be contextualized to avoid inconsistencies.

Contextualizing *beliefs* may avoid direct inconsistencies but is not sufficient to avoid conceptual conflicts nor relate possibly conflicting or partially redundant statements. E.g., a relation between the statements "according to user X, birds fly" and "according to user Y, healthy adult carinate birds can fly" is necessary to state whether the second statement is a correction (by Y) of the first statement, or if the first statement is a correction (by X) of the second statement. Such a relation can then be exploited by each user (according to preferences and application requirements) for manually or automatically selecting which statement should be exploited by an inference engine if it has to choose between the two. For knowledge retrieval purposes, a choice may not have to be made since, when the two statements are potential answers to a query, returning both, connected by their relation, may be a good and informative result. One user-specified strategy of automatic exploitation strategy may be: "when statements conflict and when their authors are all trustable, select the most corrected statements according to their inter-relations and then, if conflicts remain, give the result of the inferences for each selectable combination of non-conflicting set of statements".

The above approach is further detailed and implemented in the *shared KB editing protocol* of the WebKB-2 server [9]. It handles any KB sharing conflict via the addition of relations to the KB. E.g, terms or relations which are made obsolete by their creators but used by other agents are not fully removed but contextualized in a way that indicates i) for terms, who are their new owners, and ii) for relations, who does not believe in them anymore. Regarding additions to the KB, the main rule is that, when the addition of a statement to the shared KB would lead to a potential conflict or implicit redundancy with already stored statements, the protocol asks for the new one to *be directly or indirectly related* to each of those stored ones by a relation of one of the following types: "`pm:non-corrective_=>`", "`pm:non-corrective_<=`", "`pm:corrective_=>`", "`pm:corrective_<=`", "`pm:corrective_reformulation`", "`pm:corrective_exclusion`", "`pm:corrective_alternative`" and "`pm:statement_instantiation`". Since these relation types *either* specialize the *generalized implication relation type* "`pm:correc-`

`tive-or-non-corrective_=>`", (which is transitive) *or* its exclusion (the type "`pm:implication_exclusion`", alias "`pm:=>!`"), i) the protocol leads all the statements of the KB to be organized into a hierarchy based on this generalized relation type, and ii) all the potentially redundant or conflicting statements are (directly or transitively) connected via this generalized implication relation or its exclusion. These last two points are useful for inferences, checks and quality evaluations of the KB. Since people can use the above cited relations even when an inference engine is not able to detect potential conflicts or implicit redundancies, people may use such relations between informal statements within a KB or a semantic wiki. Thus, the approach may *also* be used to organize the content of a semantic wiki and avoid or solve edit wars in it. To conclude, this approach works with any kind of knowledge, does not arbitrarily constrain what people can represent and store (within the constraints of any automatically enforceable topic or KB scope), and keeps the KB organized, at least insofar as the used inference engine can detect inconsistencies or redundancies.

2.4 KB Servers That Support Networked KBs

As hinted in the first paragraph of the introduction, the amount of information that can be valuable for risk management is huge (and can be used for many other purposes). That information cannot be stored into a single *individual KB* (alias, *physical KB*), i.e. a KB which i) has an associated *KB server* storing it, and hence, ii) unlike a *networked KB* (alias, *virtual KB*), is not composed of a network of individual KBs exchanging information or forwarding queries via the KB servers associated with these KBs. As also hinted in the introduction, the W3C does not make any recommendation about networked KBs, it solely advises the authors of KBs to relate the terms of their KBs to those of some other KBs. Yet, the more knowledge is added to independently or semi-independently developed KBs, i) the more implicit redundancies and inconsistencies they have between them, ii) the harder it is to fix these problems, and iii) each user that wants to reuse these KBs has to (re-)do this work.

For reasons similar to those given in the previous (sub-)sections, for a networked KB to be scalable and interesting for general KS purposes, i) its total content, i.e. the sum of its component KBs, should be as organized as if it was stored in an individual shared KB with the properties described in the previous subsection, ii) neither the repartition of the KRs amongst the KBs, nor adding one's own individual KB to a networked KB, should depend on a central authority (automated or not), and iii) no user of the networked KB should have to know which component individual KB(s) to query or add to. Hence, *ideally*, i) there would exist at least one networked KB organizing all KRs on the Web, and ii) additions or queries to one KB server would be automatically forwarded to the relevant KB servers. In distributed or federated databases, the protocols that exchange information or forward queries exploit the fact that each individual database has a known and *fixed* (small) schema. These database schema based protocols cannot be directly adapted to networked KBs since the counterpart of a database schema in a KB is its ontology, which is generally large and *often modified* by the KB contributors. Many architectures advocated in disaster management related articles, e.g. those of [2] are based on – or related to those of – dis-

tributed or federated databases. Extending the classic peer-to-peer protocols by taking into account the ontologies of the component individual KBs is also not scalable: i) this either implies a "database schema based"-like solution (and then the ontologies can hardly be modified) or a centralization mechanism, and ii) this does not address potential implicit redundancies and inconsistencies between KBs.

To satisfy all the above cited constraints, the solution proposed by [8] is based on the notions of "(individual KB) scope" and "nexus for a scope". The rest of this section presents the underlying ideas of a recently extended version of this solution. An *intensional scope* is a specification (via a KR) of the kinds of objects (terms and KRs) that the server handling an individual KB is committed to accept from Web users. This scope is decided by the owner of the shared individual KB. An *intensional core scope* is the part of an intensional scope specifying the kinds of objects that a server is committed to accept even if, for each of these kinds of objects, another intensional core scope on the Web also includes this kind of objects (i.e., if at least another server has made the same storage commitment for this kind of objects). An *extensional scope* is a structured textual document that lists each formal term (of the ontology from the individual KB) using a normalized expression of the form: "<formal-term-main-identifier>__scope <URL_of_the_KB>". This format permits KB servers to exploit Google-like search engines for knowing which KBs store a particular term. A *(scope) nexus* is a KB server that has publicly published its intensional and extensional scopes on the Web, and has also specified in its non-core intensional scope that *it is committed to accept storing the following kinds of terms and KRs whenever they do not fall in the scope of another existing nexus*: i) the subtypes, supertypes, types, instances of each type covered by its intensional scope, and ii) the direct relations from each of these last objects (that are stored in this KB only as long as no other nexus stores them). (The WebKB-2 server that hosts the MSO is a nexus that has at least the MSO has intensional scope. Thus, this server can be used by any networked KB as one possible nexus for non-domain specific terms and KRs.) Then, "the *joining* of an individual KB (server) to a networked KB" simply means that the KB server is being committed not only to be a nexus for its intensional scope but also to perform the following tasks whenever a *command (query or update)* is submitted to the KB server:

- The first task is, insofar as the intensional scope allows it, to *handle this command internally* via the KB sharing protocol of WebKB-2 or another protocol with better properties. For efficiency reasons, when an object is in the core intensional scope but is related to other objects that are not in it, each of these other objects should be associated to the URL of another KB server that has this object within its core intensional scope.
- The second task is to forward this command to the KB servers which, given their scopes, may handle it, at least partly. These servers are retrieved via the above cited URLs and/or exploitation of a Google-like search engine.

Via this propagation, the commands are forwarded to all nexus that can handle them, and no KB server has to store all the terms of all the KBs, even for interpreting the published nexus scopes. To counterbalance the fact that some KR forwardings may be lost or not correctly performed, i.e. that this "push-based strategy" may not always work, each KB server may also search other nexus having scopes overlapping its own scopes

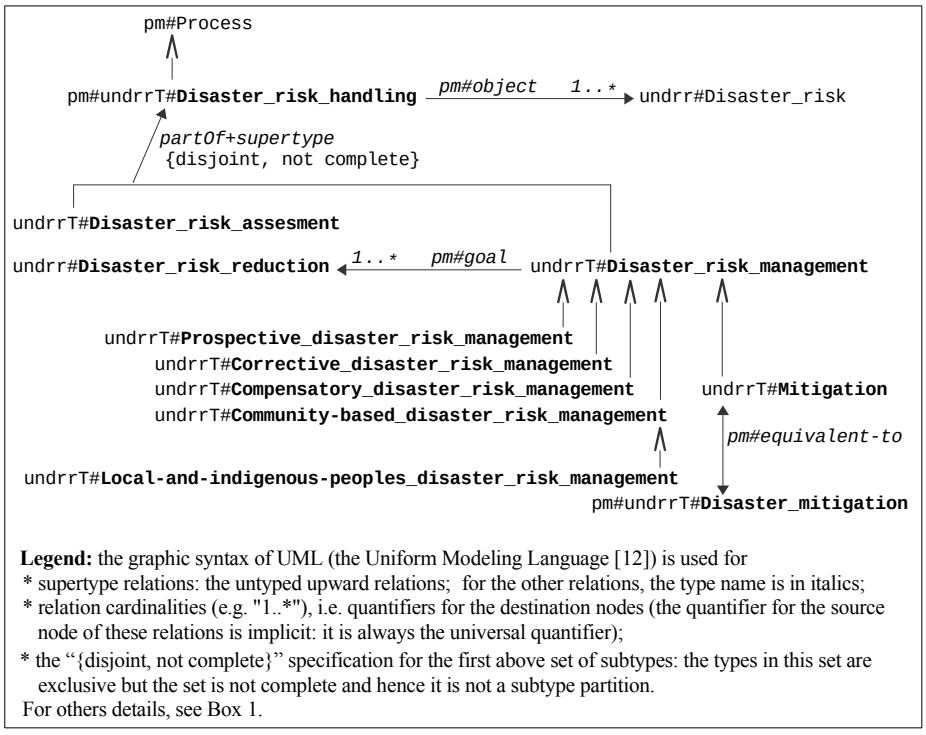
and then import some KRs from these nexus (this is the complementary "pull-based strategy"). Thus, KB servers with overlapping scopes have overlapping content but this redundancy is not implicit and hence not harmful for inference purposes.

3 Examples Of Representations For General Knowledge Sharing

In this section, for concision and clarity purposes, the FL notation [8] is used, not a W3C KRL notation. Regarding identifiers, the difference is minimal: the namespace prefixing separator is "#" (as in `pm#Table`) instead of ":" (as in `pm:Table`), since ":" is instead used to delimit relation nodes, as in most frame-based KRLs.

3.1 Organization of a Small Terminology About Disaster Risk Reduction

In 2017, the UNDRR (United Nations office for Disaster Risk Reduction) has defined a "terminology about disaster risk reduction [16]" which is here referred to as "UndrrT". [11] is a Web document that represents UndrrT via the FL KRL and, as illustrated by Fig. 1 and Box 1, organizes it into a subtype hierarchy using i) subtype partitions or



Legend: the graphic syntax of UML (the Uniform Modeling Language [12]) is used for
* supertype relations: the untyped upward relations; for the other relations, the type name is in italics;
* relation cardinalities (e.g. "1..*"), i.e. quantifiers for the destination nodes (the quantifier for the source node of these relations is implicit: it is always the universal quantifier);
* the "{disjoint, not complete}" specification for the first above set of subtypes: the types in this set are exclusive but the set is not complete and hence it is not a subtype partition.

For others details, see Box 1.

Fig. 1. UML-like representation of the relations represented with FL in Box 1

exclusions, whenever possible, ii) the top-level concept types of the MSO, and iii) a few additional types when really needed for categorization purposes. This Web document is also structured into sections and subsections according to some of the MSO types, in a systematic and non-subjective way. All these points make the *terms and relations between the terms* in UndrrT *much easier to understand and retrieve (via queries or by following relations)* than in the UNDRR document where they are listed in alphabetic order and only informally defined.

The above first and second points also support some automatic checking of the way these terms are *specialized or used in KRs*, to detect whether some of their meaning has been misinterpreted. E.g., instances of `undrrT#Disaster_risk_management` can only be *sources* of relations the signature of which has `undrrT#Disaster_risk_management` or one of its supertypes as the first parameter. Since one of these supertypes is `pm#Process`, and since the MSO provides many types of relations from `pm#Process` (e.g. `pm#object`, `pm#parameter`, `pm#duration`, `pm#agent`, `pm#experiencer`, etc.), these relations can be used from instances of `undrrT#Disaster_risk_management`.

Representing UndrrT via the MSO also highlighted important ambiguities that the sometimes lengthy informal definitions associated with the terms did not help resolve. E.g., are the types `undrr#Exposure`, `undrr#Vulnerability` and `undrr#Resilience` supposed to be subtypes of `pm#Characteristic_or_dimension_or_measure` or of `pm#State`? In [11], the first option has been chosen because it eases the use of such types in KRs but other users of UndrrT may have used such terms as if they represented states. The two interpretations are exclusive: they cannot be reconciled. Thus, such ambiguities clearly limit general KS.

Box 1. Commented extract of the FL representation of the UNDRR terminology (same content as in Fig. 1: this extract does not include relations for informal definitions and annotations but here has many comments explaining the meaning of the used abbreviations and FL expressions)

```
//Comments are prefixed by "//" and here in italics; the FL namespace separator is '#', not ':'.
pm#undrrT#Disaster_risk_handling  //"pm#undrrT#": the type, created by pm, was implicit in UndrrT
  /^  pm#Process,  //"/^" or "  ": supertype relation in FL
  pm#object: 1..* undrr#Disaster_risk,  //"1..*": one or several
  \.part:  //"subtype relation" and "part relation between the instances of the connected types"
    e{ //In addition to be destinations of "\.part", the next two types are exclusive: "e{...}"
      undrrT#Disaster_risk_assesment
      (undrrT#Disaster_risk_management  //"(...)": isolation of relations starting from this type
        pm#goal: 1..* (undrrT#Disaster_risk_reduction
                       pm#parameter: 0..* undrrT#Disaster_risk_reduction_strategy_or_policy ),
        \. //"\." or "  ": subtype relation in FL
          //No "e{ ...}" here since the following subtypes are not necessarily exclusive
          undrrT#Prospective_disaster_risk_management    //This type and its next four siblings
          undrrT#Corrective_disaster_risk_management     //  are direct subtypes of
          undrrT#Compensatory_disaster_risk_management   //  undrrT#Disaster_risk_management
          (undrrT#Community-based_disaster_risk_management
            \. undrrT#Local-and-indigenous-peoples_disaster_risk_management )
          (undrrT#Mitigation  //Since this type name is ambiguous, pm adds a clearer one
            = pm#undrrT#Disaster_mitigation           //  via this equivalence relation
          ) __[author: pm]  //pm believes that the last subtype relation is true even though
                            //  it is not in UndrrT (neither explicitly not implicitly)
    )  //End of relations from undrrT#Disaster_risk_management
  }.  //End of the exclusion set and of all relations
```

3.2 A General Model To Represent And Organize Search&Rescue Information

Unlike other general ontologies, the MSO provides a type for "description instruments or results" (alias, "information objects", e.g. languages, procedures, object-oriented classes) *and* many subtypes for it, most of which are from KRLO. They are useful for representing and categorising many information objects in disaster management. Box 2 shows some types that are useful in the *Search & Rescue* domain.

Box 2. Subtype hierarchy of MSO types that are useful for categorizing description-related types in *Search & Rescue* representations

```
//For clarity purposes, an informal representation is used below, not a representation in FL:
//  an indented list is used for showing subtype relations between types,
//Still for clarity purposes, from now on in boxes and figures, the source prefix of each
//  type identifier is left implicit (-> all types come from the MSO).
//Below, in this box, bold characters are used for referring to terms that are listed in Box 3.

Description_instrument-or-result-or-container
  Description_instrument-or-result              //Alias Information_object
    Semantic-representation_instrument-or-result //E.g. Principle_of_Coriolis_acceleration
      Semantic-representation_instrument        //E.g. Java_semantic, Logic_semantic, Type
      Semantic-representation_result            //E.g. Semantic_of_a_KB, Semantic_of_a_program
    Structural_representation_instrument-or-result
      Abstract_representation_instrument-or-result
        Abstract_representation_instrument
          Abstract_process-structure            //E.g. While_loop, Petri-Net_structure_element
            Abstract_function                   //E.g. each method in Box 3
          Abstract_data_type                    //E.g. Object_oriented_class, Array, Integer
          Abstract_representation   //E.g. Path_representation and each term in bold in Box 3
          Abstract_language-or-language_element  //E.g. Java_abstract_grammar
        Abstract_representation_result          //"Abstract_representation_instrument instance"
          Search_algorithm
            Graph-traversal_and_path-search_algorithm //E.g. the A* algorithm
      Concrete_representation_instrument-or-result
        Concrete_representation_instrument      //E.g. Java_concrete_grammar, Character
        Concrete_representation_result          //E.g. Java_concrete_function
  Description_container                         //E.g. File, Software, Web_server, KB_server
```

A concrete map, e.g. on a screen, is a 2D or 3D graphic representation of physical objects. An abstract map is a structural representation of a concrete map. In *Search & Rescue*, search functions need to exploit characteristics of objects in a map, and search agents that do terrain mapping or discover victims or possible indices of victims need *to add* objects to the map. Thus, structurally, an abstract map *should not be a set of pixel representations* but should permit the storage, update and querying of i) object representations that *are, were or may be part of* the map, and hence also ii) at least their `partOf` relations, types and attributes. This does not mean that such maps should be fully represented *using relations*, in a KB. Indeed, this would not only be an inefficient way to store and handle spatial coordinates or relationships of objects in maps, this would also make them difficult to exploit via classic programs, those based on classic structures such as object-oriented classes. Hence, such maps should remain *abstract data structures* but should be represented or implemented in *much richer structures* than "Simple Vector Graphics (SVG) or 2D/3D arrays" (the kinds of structures used in current abstract maps). *Ideally*, for each physical object, such maps would store identifiers or pointers of the object *in a KB*, and this KB would support semantic queries on the physical objects. Box 3 contains a generic representation of all such abstract maps useful for *Search*

Box 3. Commented FL representation of object-oriented classes for *Search & Rescue*

```
//The types in bold characters (in italics or not) are Abstract_representation types. The types in
//  in italics (and not in bold) are information object types that are not Abstract_representation types.
//The other types (except for "Thing") are subtypes of Characteristic_or_dimension_or_measure.
//Variable names are prefixed by "?", as in many other KRLs.
//As in the previous boxes, when comments at the right of some code line are spread on multiple lines,
//  each expression in a line is mostly focused on the code of that line.

Abstract_map  /^ Abstract_representation,      //Representation of a class for maps
 _{ attribute: 1 Map_scale,          //The scale of a map should be associated to it
               1 Temporal-point-or-region_coordinate ?timeStamp,   //When the map was valid
               1..3 Spatial-point-or-region_coordinate;            //A 2D/3D point/area
    part: 1..* Physical_object_representation_in_an_abstract_map;  //Object parts
          //This set can be implemented via a 2D/3D array or an SVG structure
    method: Abstract_map__objects_possibly_at      //------ For retrieving objects in (a portion of) a
           (1 Abstract_map, 1..3 Spatial-point-or-region_coordinate,          //  map (specified here),
            0..* Type ?typeOfAtLeastOneOfTheSearchedPhysicalObjects,          //  wrt. their types
            0..* Attribute ?attributeOfAtLeastOneOfTheSearchedPhysicalObjects) //  or attributes, e.g.
                 //  health, social value, etc. The next line specifies the types in the returned set
           -> .{1..* Physical_object_representation_in_an_abstract_map};     //-> The retrieved objects
    method: Abstract_map__values_of_objects_possibly_at     //------ For knowing the values of objects
           (1 Abstract_map, 1..3 Spatial-point-or-region_coordinate,    //  in (a portion of) a map
            0..* Type ?typeOfAtLeastOneOfTheSearchedPhysicalObjects,    //  given the types&attributes
            0..* Attribute ?attributeOfAtLeastOneOfTheSearchedPhysicalObjects, //  of searched objects
            1..* Temporal-point-or-region_coordinate ?valuesDuringThisTimePeriod,  // at a given time,
            0..* Environmental_context ?environmentalContextOfTheSearch)  //  wrt. the weather, ...
           -> .{0..* Representation_of_the_value_of_a_physical-object};   //-> The retrieved values
    method: Abstract_map__best_paths_from_somewhere_to_at_least_1_object //------ For knowing the best
           (1 Abstract_map                                          //  paths to take (in a map),
            1..3 Spatial-point-or-region_coordinate ?fromPlace,          //  from a place to
            1..3 Spatial-point-or-region_coordinate ?regionOfSearchedObjects,  //  another, to find
            0..* Type ?typeOfAtLeastOneOfTheSearchedPhysicalObjects,           //  objects of given
            0..* Attribute ?attributeOfAtLeastOneOfTheSearchedPhysicalObjects, //  attributes, at
            1..* Temporal-point-or-region_coordinate ?valuesDuringThisTimePeriod,  //  a given time,
            0..* Environmental_context ?environmentalContextOfTheSearch,  //  wrt. the weather, ...,
            0..* .{Thing, 1..* Type ?typeOfAttributeOfTheThing,       //  given constraints on the
                          0..1 Value ?maxValue, 0..1 Value ?minValue  //  types+values of the objects
                 } ?constraintsDuringTheSearch,                       //  to find, while minimizing
            0..* Type ?typeOfAttributeToMinimizeForBestPaths,  //  some attributes (e.g. Battery_use)
            0..* Type ?typeOfAttributeToMaximizeForBestPaths,  //  & maximizing others (e.g. Safety)
            0..1 Abstract_function ?fctToSelectBestPaths,      //  and/or using a function to do so;
            0..1 Integer ?MaxNumberOfBestPaths,                //  a maximum number of best paths and
            0..* Search_algorithm ?preferredSearchAlgorithm)   //  a given algorithm may also be used
           -> 0..* .{1..* Spatial-point-or-region_coordinate};  //-> The computed best paths
 }.

Physical_object_representation_in_an_abstract_map
 _{ attribute: 0..1 Reference_to_a_semantic_representation, //Identifier of (or pointer to) a KB object
                                                           //  that represents this physical object
              1 Representation_of_the_location_of_a_physical-object,
              0..* .{ 1 Physical-object_attribute, 0..1 Certitude_of_a_value };
    part: 0..* .{ 1 Physical-object_representation_in_an_abstract_map ?embeddedObject,
                  0..1 Certitude_of_a_value };
    part of: 0..* .{ 1 Physical-object_representation_in_an_abstract_map ?embeddingObject,
                     0..1 Certitude_of_a_value };
    method: Physical_object_representation_in_an_abstract_map__value
           (1 Physical_object_representation_in_an_abstract_map,
            1..* Temporal-point-or-region_coordinate ?valueDuringThisTimePeriod,
            0..* Environmental_context_of_a_search)
           -> 1 Representation_of_the_value_of_a_searched_physical-object
 }.

Representation_of_the_location_of_a_physical-object
 _{ attribute: 1..3 .{ 1 Spatial-point-or-region_coordinate, 0..1 Certitude_of_a_value }  }.

Representation_of_the_value_of_a_physical-object
 _{ attribute: 1 Quantitative-or-qualitative_social_value_of_something, 1 Certitude_of_a_value  }.
```

& Rescue, i.e. a list of relations between such maps and some other kinds of objects. This representation can be viewed as a generalization or "minimal general specification" of *abstract data structures* for such maps. More comprehensively, Box 3 is a top-level ontology – hence a minimal general specification, model or listing – of functions, and of the most interesting kinds of objects that they could exploit, which are useful for *Search & Rescue*. Three important and combinable functions are represented:

- one to *retrieve objects* (generally, *people*, but this is irrelevant for a generic specification) in (a portion of) a map, given some of their types or ranges for their attributes, e.g. a range for the expected health or social value of actual or potential victims at certain places in a map (since, for example, an often used strategy is to first try to save the healthier and most socially valuable victims),
- one to *compute values* (possibly with some associated certitude coefficients) for particular attributes of particular objects in a map, given other parameters such as the environmental context (weather, ...) and when the rescue begins and/or when the objects can or could be retrieved (since, for example, some victims may be difficult to save by the time they are found),
- one to *compute the best paths* (possibly given strategic rules and/or a search algorithm) *from a starting place to others* (this may be a whole area to explore) *for finding objects of given attributes*, with additional attributes to maximize (e.g. the safety of the rescuing agents and of the victims) and others to minimize (e.g. the power consumption of a rover used for exploring a disaster area).

In object-oriented (OO) programming, functions are often associated with some of the objects they exploit by being represented as methods of the class of these objects. This kind of association helps organizing and normalizing the code, and is mainstream. Since this association is worth representing and since the content of Box 3 is intended to be a minimal general specification of important primitive functions for *Search & Rescue*, FL has recently been extended to allow the use of the syntax used in Box 3. This syntax is not only close to those of frame-based or graph-based KRLs, including Turtle and JSON-LD, but also *close to UML notations and OO-like notations*. The use of "_{" and "}" as delimiters permits the use of this syntax which slightly departs from the usual one in FL (but FL often uses "." and "_" as prefixes for specifying particular meanings, e.g. ".{" and "}" are the default delimiters for collections in FL). However, the represented KRs are not just OO classes, if only because genuine relations are used, not attributes of classes that are local to them. Here are two advantages of associating functions to information structures via a relation of type "method":

- This allows the use of an intuitive and compact OO-like *naming scheme* for functions: see the "__" within the names where OO programming languages would allow the use of "." for compactness and modularity reasons.
- Combined with the use of UML cardinalities (e.g. "1..3", "0..*") in the parameters of these functions or methods, such a graph-based specification clearly generalizes – or abstracts away – implementation particularities. Indeed, with programming languages, the structures (or *classes* in OO languages) are tree structures, and the functions do not use cardinalities nor have successive default parameters; this generally forces the users of these languages to i) cut the semantic graph into pieces when representing it via

such structures, ii) make the relations *implicit*, iii) choose a rather arbitrary embedding order between the graph elements, and iv) implement various similar versions of a same function, based on particular aggregations of datatypes for the parameters.

3.3 Representations About Automatic Explorations of A Disaster Area

This subsection shows how the task of *systematically exploring* a disaster area (e.g., by a rover, to search for victims) can be represented *at a high-level*: the reuse of the functions from the previous subsection is not shown. The focus is to show how tasks or procedures can be represented *via KRs*, at a high-level.

Box 4 shows an example systematic search procedure written in a procedural language. Such procedures can most often be automatically converted into pure functions, hence in a declarative way. Such functions can be directly represented in a KRL that handles functions (e.g. KIF and FL) and hence included in a KB. There, functions can be organized via generalization relations and also generalized by more classic KRs, e.g. logical formulas representing rules. Box 4 is just an example procedure. Fig. 2 (next page) shows some relations (a partOf one and several subtype ones) between top-level tasks in *Search & Rescue*. Such relations are useful for categorization purposes, e.g. to organize and exploit a library of functions useful for *Search & Rescue*. Such a library may for example organize functions which represent different ways of performing similar tasks. The library – and hence programs reusing it – may also include a function that selects the most relevant of these different ways for a particular environmental context given as a parameter. Box 5 shows some further subtype relations from one of the tasks mentioned in Fig. 2 and, to do so, uses FL.

Box 4. Commented procedure for a systematic search by a rover, one based on an infinite loop in which the rover simply decides to go ahead or not

```
while ( true )  //Infinite loop. Below, "()" indicates a function call (the parameters are not specified)
{ if  ( further_exploring_is_not_useful() )  //To decide that, the methods of Section 3.2 are used
  { come_back_to_base();  break; } //"break": the loop is broken when the rover has returned
  else if ( going-ahead-and-then-come-back-to-base_is_not_possible() )  //Via the methods of Section 3.2
         come_back_to_alternative_route (); //E.g., given battery levels, obstacles, mechanical problems
       else  go_ahead();
}
// Two example cases for a rover exploring underground spaces and fails, under debris and ruins:
// * The rover cannot continue on a particular path (e.g. because it would risk getting stuck):
//   it returns in the opposite direction to a point where it can continue its exploration,
//   an intersection with a not yet explored path.
// * The rover has explored the last path (-> "normal" end of mission) or
//   cannot continue exploring (e.g. because it has not enough energy): it returns to its base.
```

Box 5. FL categorization of the "Safe_path_backtracking" process or task mentioned in Fig. 2

```
Selecting_a_path /^ Process,          //reminder: here, only type names are used (not type identifiers)
 part of: 0..* (Search_and_rescue /^ Process),
 \. (Selecting_a_safe_path \. (Selecting_a_safe_and_recently_explored_path \. Safe_path_backtracking) ),
 \. partition
    { Path_selection_when_going_ahead_is_possible_and_useful
      (Path_selection_when_going_ahead_is_not_possible_or_not_useful \. Safe_path_backtracking)
    }.
```

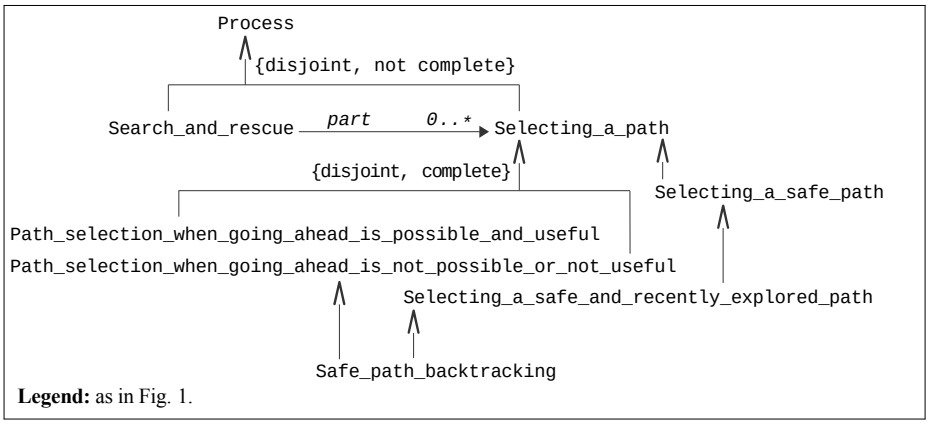


Fig. 2. UML-like representation of some relations between tasks involved in *Search & Rescue*

4 Conclusion

The first kinds of contributions of this article were i) its highlighting of the insufficiencies of *restricted KS* – hence, the waste of efforts and opportunities that not using *general KS* is for supporting general tasks such as disaster management – and ii) its panorama of complementary techniques supporting *general KS*. Since this field is still seldom researched, for the techniques to be complementary, this panorama draws on techniques previously developed by the first author. Although some new research elements have been included, the originality of this panorama is mainly in the *synthesis* it makes since the presented techniques together provide a rather complete approach for supporting general KS efforts useful for disaster management, while still allowing the reuse of advances in the well researched field of restricted KS. Together, these techniques answer the following research question: how to let Web users collaboratively build KBs i) that are not *implicitly* "partially *redundant* or *inconsistent*" internally or with each other, ii) that are complete with respect to certain criteria or subjects, iii) without restricting what the users can enter nor forcing them to agree on terminology or beliefs, and iv) without requiring people to duplicate knowledge in various KBs, or to manually search knowledge in various KBs and aggregate knowledge from various KBs?

The second kinds of contributions of this article were i) KRs showing how complementary kinds of disaster management related information can be represented for general KS purposes, and ii) highlights of the interest of creating or reusing such KRs. The focused example domains were i) the UNDRR terminology, ii) a general model to represent and organize *Search & Rescue* information, and iii) procedures or tasks for automatically exploring a disaster area.

The authors of this article will continue to add KRs to the MSO of the WebKB-2 server for supporting disaster management. WebKB-2 will continue to be improved to ease its use for general KS in disaster management.

References

1. Dodds, L., Davis, I.: Linked Data Patterns – A pattern catalogue for modelling, publishing, and consuming Linked Data. Web doc. (56p): http://patterns.dataincubator.org/book (2012).
2. Dobrinkova, N., Kostaridis, A., Olunczek, A., Heckel, M., Vergeti, D., Tsekeridou, S., Seynaeve, G., De Gaetano ,A. Finnie, T., Efstathiou, N., Psaroudakis, C.: Disaster Reduction Potential of IMPRESS Platform Tools. In: revised selected papers of ITDRR 2016, pp. 225-239, Springer, Cham. (2016).
3. Gaur, M., Shekarpour, S., Gyrard, A., Sheth, A.: empathi: An ontology for emergency managing and planning about hazard crisis. In: 2019 IEEE 13th International Conference on Semantic Computing (ICSC), pp. 396-403 (2019).
4. Kontopoulos, E., Mitzias, P., Mossgraber, J., Hertweck, P., van der Schaaf, H., Hilbring, D., Lombardo, F., Norbiato, D., Ferri, M., Karakostas, A., Vrochidis, S., Kompatsiaris, I.: Ontology-based Representation of Crisis Management Procedures for Climate Events. In: ICMT 2018 (Workshop on Intelligent Crisis Management Technologies for Climate Events), at ISCRAM 2018, Rochester NY, USA (2018).
5. Malizia, A., Astorga-Paliza, F., Onorati, T., Díaz, P., Aedo Cuevas, I.: Emergency Alerts for all: an ontology based approach to improve accessibility in emergency alerting systems. In: ISCRAM 2008, pp. 197-207, Washington DC, USA (2008).
6. Martin, Ph.: Knowledge representation in CGLF, CGIF, KIF, Frame-CG and Formalized-English. In: ICCS 2002, 10th International Conference on Conceptual Structures, LNAI 2393, pp. 77-91 (2002).
7. Martin, Ph.: The Multi-Source Ontology (MSO) of WebKB-2. Web document: http://www.webkb.org/doc/MSO.html (2004).
8. Martin, Ph.: Towards a collaboratively-built knowledge base of&for scalable knowledge sharing and retrieval. HDR thesis (240 pages; "Habilitation to Direct Research"), University of La Réunion, France, http://www.webkb.org/doc/papers/hdr/ (2009).
9. Martin, Ph.: Collaborative knowledge sharing and editing. International Journal on Computer Science and Information Systems (IJCSIS), 6(1), 14-29 (2011).
10. Martin, Ph., Bénard, J.: Creating and Using various Knowledge Representation Models and Notations. In: ECKM 2017, pp. 624-631, Barcelona, Spain (2017).
11. Martin, Ph.: Representation and organization of the UNDRR terminology. Web document: http://www.webkb.org/kb/nit/o_risk/UNDRR/d_UNDRR.fl.html (2020).
12. OMG (Object Management Group): OMG Unified Modeling Language Superstructure Specification, version 2.1.1. Document formal/2007-02-05, Object Management Group, http://www.omg.org/cgi-bin/doc?formal/2007-02-05 (2007).
13. Munkvold, E.B., Opach, T., Pilemalm, S., Radianti, J., Rod, J.K.: Sharing Information for Common Situational Understanding in Emergency response. In: European Conference of Information Systems (ECIS), Uppsala, Sweden (2019).
14. Snaprud, M., Radianti, J., Svindseth, D.: Better access to terminology for crisis communications. In: revised selected papers of ITDRR 2016, pp. 93-103, Springer (2016).
15. Tanzi, T.J., Bertolini, M.: 3D Simulation to Validate Autonomous Systems Intervention in Disaster Management Environment. In: revised selected papers of ITDRR 2019, pp. 196-211 (16 pages), Springer, Cham. (2019).
16. UNDRR (United Nations Office for Disaster Risk Reduction): Report of the open-ended intergovernmental expert working group on indicators and terminology relating to disaster risk reduction. Web document: https://www.preventionweb.net/publications/view/51748 (2020).
17. W3C (World Wide Web Consortium): Semantic Web. Web document: https://www.w3.org/standards/semanticweb/ (2020).