sustainability

MDPI

Article

General Knowledge Representation and Sharing, with Illustrations in Risk/Emergency Management

Philippe A. Martin [1,*] and Tullio J. Tanzi [2]

[1] EA2525 LIM, I.T. Department, University of La Réunion, 97400 Saint-Denis, France
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris, 91764 Palaiseau, France; tullio.tanzi@telecom-paris.fr
* Correspondence: philippe.martin@univ-reunion.fr

Abstract: Many decision-making tasks, including the sustainability-oriented ones and those related to the management of risks or emergencies, must gather, integrate, and analyze an important amount of information of various kinds and origins. Hence, how *should* information be best organized and shared by agents – people or software – for *all and only* the pieces of information looked for by these agents to maximize their retrieval, reuse, organization and analysis by these agents? To that end, various *logic-based* knowledge representation (KR) and sharing (KS) techniques, and hence KR bases, have been used. However, most KS researchers focus on what this article defines as "*restricted* KR and KS", where information providers and consumers can or have to discuss for solving information ambiguities and other problems. The first part of this article highlights the usefulness of "*general* KR and KS" and, for supporting them, provides a *panorama* of complementary techniques, and hence, indirectly, best practices or kinds of tools to use for general KS purposes. These techniques *collectively* answer research questions about how to support Web users in the collaborative building of KR bases. The second part uses the risk/emergency management domain to illustrate the ways different types of information can be represented to support general KS.

Keywords: ontology sharing; knowledge sharing; knowledge representation; risk management

Citation: Martin, P.A.; Tanzi, T.J.; General Knowledge Representation and Sharing, with Illustrations in Risk/Emergency Management. *Sustainability* **2023**, *15*, 10803. https://doi.org/10.3390/su151410803

Academic Editors: António Abreu, Jie Jiang, Sisi Zlatanova, Edoardo A. C. Costantini and Yasuhide Hobara

Received: 22 June 2022
Revised: 29 July 2022
Accepted: 5 August 2022
Published: 10 July 2023

1. Introduction

Many tasks first require retrieving, comparing, aggregating and *organizing* an important amount of information of many different kinds in order to make good and timely decisions. This is the case of sustainability-oriented decisions, if only because they have to balance economical, societal and environmental issues. This is also the case of many tasks for the management of risks or emergencies. E.g., both Search&Rescue and preemptively reducing disaster risks require access and use of many kinds of information or other resources, such as particular kinds of persons, detection devices, communication tools, maps, search methods and search software. These tasks also depend on many parameters such as the nature of the emergency, the weather, the terrain and the availability of the needed resources.

Ideally, to support such tasks and hence the findability, gathering, interoperability, reusability, integration and analysis of information potentially useful to those tasks or to the design of tools for those tasks, that information should be published, related and organized on the Web in places and in ways that allow people and software agents to (i) retrieve and compare information with respect to non-predefined sets of criteria, and (ii) complement information while keeping them as organized and hence as retrievable.

As explained below, *one* requirement for such an ideal and scalable organization – and thus a *primary very general best practice* for information dissemination and collaboration between people, organizations or software – is to represent and organize information either directly within *knowledge representation* bases (*KR bases*) or in ways that can be automatically imported into KR bases (e.g. in documents and databases that have been designed to allow such an importation). These KR bases can be either privately developed or, preferably, collaboratively developed.

In this article, these KR bases are simply called *KBs* and, before going further, need to be more introduced now. Such KBs do not store texts or other *data*; they store KRs (or simply, "knowledge"), i.e. logic-based representations of *semantic* relations between pieces of information – *semantic* relations being relations that *can* be represented in a logic-based way. The boxes and figures in Section 2.1 and Section 3 include many examples. In this article, the notions referred to by the words "*knowledge*" (KRs) and "*data*" are mutually exclusive. "*Data*" refers to information not explicitly organized – or poorly organized – by semantic relations, e.g. as in databases or XML documents: they are mainly organized by predefined *structural* relations (i.e. partOf ones) and few *semantic relations of very few predefined types* (mostly typeOf relations and sometimes subtype relations). In KBs, unlike in relational databases, all the types (i.e. relation types and concept types) and their definitions are user-provided (not predefined by the database designer); most of the knowledge in many KBs are expressed via such definitions; large KBs such as CYC [1,2], Freebase [3] and DBpedia [4] have hundreds of thousands of subtype relations. Document-based technologies and database systems generally only handle *data*, although deductive databases may be seen as steps towards KBs. A KB is composed of an ontology and, generally, a base of facts. An ontology is (i) a formal terminology, i.e. a set of terms (alias, object identifiers) used in the representations stored in the KB, along with (ii) representations of term definitions, and thereby direct or indirect semantic relations between these terms. Databases and natural-language-based documents cannot automatically be converted into *KBs that are well-organized via generalization and implication relations*, if only because these documents and bases most often lack the necessary information to derive such relations (these relations are rarely made explicit by document authors and even human readers often cannot infer such relations with certainty). These relations – and thus, manually or semi-automatically built KBs – are necessary for the support of (i) *semantic*-based searches, via queries or navigation, and (ii) any *scalable* way of integrating or organizing information. This explains why architectures or methodologies for building ontologies or systems exploiting them have already often been discussed regarding disaster risk reduction or management. For example, in February 2022, the digital library of the ISCRAM conferences ("Information Systems for Crisis Response and Management" conferences) included 64 articles with main fields mentioning ontologies, and 46 of these articles recorded "ontology" as a keyword.

Several small *top-level* ontologies related to disaster risk reduction or management, e.g. the agent-oriented ontology of [5] for better indexing and retrieving "disaster management plans" in document repositories for such plans, SEMA4A [6] which supports alerting people about imminent disasters, *empathi* [7] which is more general and integrates some other ontologies, and POLARISCO [8] which is a suite of ontologies formalizing and relating the terminologies and methods of various emergency response organizations (e.g. fire departments, police, and healthcare services). However, as of 2022, it seems there are no public large *content ontology* related to disaster risk reduction or management, let alone KBs where people or organizations could relate or aggregate information. As an example, even though [9] (which is also about disaster related terminologies) mentions past "massive efforts e.g. in European projects such as DISASTER (cordis.europa.eu/project/id/285069 (accessed on 7 August 2022)), SecInCoRe (cni.etit.tu-dortmund.de/research/projects/secincore (accessed on 7 August 2022)), EPI (www.episecc.eu (accessed on 7 August 2022)), or CRISP (cordis.europa.eu/project/id/607941/reporting/fr (accessed on 7 August 2022))", the results of those projects were not KBs but reports about then planned works as well as advocated architectures or small models (top-level ontologies). There currently exist some large projects, such as the Norwegian INSITU (Sharing Incident and Threat Information for Common Situational Understanding) project (2019–2022) [10], which focus on harmonizing terminologies or on tools for the *collaborative synthesis* of information in classic media (databases, textual documents, maps, ...), not via KBs. The use of classic media make the harmonization of terminologies useful for supporting lexical searches (i.e. those proposed by current Web search engines and document editors; these are not semantic search tools). However, such an harmonization is a complex task which requires committees (hence an hierarchy-based decision-making organization) and it is useful only when its guidelines are followed (something that is not easy to do). Via KBs, harmonizing terminologies is not necessary since relations of equivalence or generalization between terms within KBs or across KBs can be added in a decentralized and incremental way by each provider of terms or knowledge. Tools that

exploit these particular relations can allow users and knowledge providers to choose the terms they wish, without this decreasing knowledge retrievability.

This article distinguishes two meanings for "knowledge sharing" (KS). The one here called "*restricted KS*" is closer to *data(base) sharing*: it is about (i) easing the exchange of structured information (KRs or structured data) between *particular* agents (persons, businesses or applications) that *can* discuss with each other to solve ambiguities or other problems, and (ii) the *complete or efficient* exploitation of the information by these particular agents, for particular applications. The other meaning, here called "*general KS*", is about people relating or representing information within or between KBs in ways that maximize the retrievability and exploitation of the information by *any* person and application. Examples of early landmark works related to general KS were Ontolingua (server, ontologies and vision) [11] and the still on-going above-cited CYC project. These two meanings are unfortunately very rarely distinguished, even indirectly, e.g. by the World Wide Web Consortium (W3C). With respect to KS, the W3C has a "Semantic Web vision" [12] of a "Web of *Linked data*" [13]. As the use of the word "data" may suggest, and as explained in Section 2, the techniques and vision proposed for these Linked Data are mainly focused on restricted KS. Indeed, the W3C had to focus on the basics and *convince industries* of the interests of KBs over databases. However, after 1997 – the beginning of the popularization of the W3C visions and languages – KS was mainly learned about and operationalized via the W3C documents and languages, and thus almost all research works in KS were implicitly in restricted KS. Among research articles related to risk or emergency management and that advocate using KBs, most rely on the W3C techniques or approach – e.g. the articles of [14] (about ontology-supported rule-based reasoning), of [15] (about ontology-supported access to particular databases) and of [16] (about a small ontology mainly including 38 concept types and 21 subtype relations, about some crisis management procedures). Previous studies into risk/emergency management have not addressed general KS in these domains and are insufficient to address the distributed and large number of potentially useful sources of 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.

When applied to programming – or, more generally, knowledge modeling and exploiting *processes or techniques as well as rules or constraints (or data structures for them) – restricted KS means* representing them (i) in a KB directly usable by a KR-based software for a particular application, *or* (ii) in a KB from which a particular program can be manually or semi-automatically generated (this is model-based design/programming). *With general KS,* these process-related resources are represented and organized into an ontology where general logical specifications are incrementally (and cooperatively) specialized by more and more precise or restricted specifications, according to selected paradigms (e.g. logical, purely functional and state-based) and their associated primitives from particular logics and languages. Since these primitives can be defined or declared in an ontology, this one can store and organize representations that are directly translatable in particular formal languages such as programming languages. Thus, if software components are stored in the lower levels of such an ontology, this one may also be used as a scalable library of software components in various languages. Via the systematic use of specialization relations and the explicit representation of any implementation choices, general KS allows the representation of specifications that are language dependent or application dependent while still maximizing knowledge reuse and thus allowing knowledge users (not just knowledge providers) to make such choices.

Knowledge representation and sharing (KR&S) – or, *a fortiori*, general KS – and the exploitation of its results has various advantages for risk/emergency management. Before an emergency occurs, i.e. in the anticipation phase, KR&S helps finding, organizing and analyzing resources (e.g. information for/on risk/emergency management techniques), designing tools (e.g. KB-based or not software and disaster area exploration robots) and testing them (e.g. via simulations). During an emergency, KR&S helps finding and coordinating resources (e.g. information and people). After an emergency, KR&S helps in organizing and analyzing data collected during the emergency (e.g. data collected by the robots) and exploits it for validating or refining hypothesis, techniques, simulation data and tools, thus for generating new knowledge. All these "KR&S helps or supports for risk/emergency management" derive from the knowledge integration and inferences they permit, compared to data-based technologies. Thus, in that respect, the helps and supports provided by KR&S technologies (such as those of data-based technologies) are not dependent on the context, e.g., earthquakes,

fires, floods, volcanic eruptions, etc. What changes depending on the context or domain is the knowledge that is represented, searched, retrieved and exploited, as well as particular features required for that, such as particular kinds of KR construct, logic or expressiveness, e.g. for spatial, temporal or probabilistic KR. The provided KR&S helps are better with general KS – hence with the techniques provided in this article – than with restricted KS since general KS (i) supports a better integration of knowledge by more people, hence more knowledge sources, and (ii) supports *each* knowledge provider, consumer or application in selecting, extending or creating the above-cited particular features they require. Finally, regarding the context independence of the panorama of techniques provided in this article, it should also be noted that these techniques were developed by the first author due to some clear insufficiencies of existing KR&S technologies for general KS, *in any domain*. The next paragraph lists these insufficiencies.

Representing rules or filling data acquiring forms for a particular application – or building a tool to support this – is different to representing knowledge for general KS purposes – or building a tool to support this, e.g. for allowing experts or companies in a particular domain to represent (in a shared KB) the products, services or knowledge they can provide, or for allowing researchers, lecturers and engineers to represent and integrate their knowledge in this shared KB for pedagogical or cooperation purposes. When representing knowledge for general KS purpose, some technological gaps in existing KR&S technologies often become apparent. First, starting from the most immediately apparent: reusing an existing large shared lexical ontology is necessary since otherwise every knowledge contributor would have (i) to define each term (word sense) and its generalizations, and (ii) relate each of them to each other term of each other contributor; in other words, they would each have to spend months or years creating and relating their own large shared lexical ontologies. Second, extending the used KR language appears useful because it is almost never expressive or concise enough to allow entering all the particular required knowledge for the particular domain to represent. Third, for representing such an amount of complex knowledge, textual KR languages are much easier to use than graphical interfaces, in the same way that, for medium-to-large programs, textual programming languages are easier to use than graphical ones. Fourth, the used KR languages allows many ways to represent equivalent knowledge but the associated inference engines are not able to find the results equivalent. Fifth, separately-built KBs – hence poorly related KBs that are often inconsistent and implicitly redundant with each other – are not exploitable for general KS: they do not contain enough information for an inference engine to integrate them reliably (analogously, a person cannot integrate texts written in a language he does not understand). Thus, general KS requires shared KBs with a KB editing protocols that (i) ensures that enough information is provided to reach and maintain a particular minimal organization in each shared KB, and (ii) does not restrict the knowledge the users want to enter in a KB as long as it is within the scope of this KB. Sixth, in addition to this inner-KB KS protocol, there is a need for inter-KB KS protocols since no single individual shared KB can host and efficiently manage all knowledge in all domains, or have a KB editing protocol that satisfy all knowledge contributors. Although identifying these problems is not too difficult when representing knowledge for general KS purpose, research avenues for solving them were original and ambitious: the work of developing and implementing all the underlying techniques, tools and general ontologies is difficult and very long. Section 2 introduces complementary techniques for supporting general KS – and hence the ideal described in the second paragraph of this introduction – via four subsections, one for each of the following four complementary topics of such a support: KR language instruments, KR content instruments (reusable ontologies; this is the topics on which most general KS related research focus), inner-KB content organization, inter-KB content organization. While doing so, Section 2 also gives (i) various rationale for the above-cited insufficiencies of classic techniques, and (ii) the constraints (or most important features to support) that explain why the provided solutions are proposed as answers to these insufficiencies. The originality of Section 2 is in the panorama or synthesis itself, rather than in the depth of the description of the introduced or cited techniques, since the first author has previously published on several of these techniques but separately, not together. However, in Section 2 some new elements are also introduced. Furthermore, the panorama shows that it is only together that these complementary techniques support general KS by collectively answering the following research question: how to allow Web users to collaboratively build KBs where pieces of information (i) are not implicitly "partially redundant or inconsistent", neither

internally nor with each other, (ii) are complete w.r.t. particular criteria and subjects selected by the KB creators, (iii) do not restrict the knowledge that people can provide nor force them to agree on beliefs or terminology, (iv) do not lead knowledge providers to duplicate information in various KBs, and (v) do not require people to search information in several KBs nor aggregate information from several KBs?

Via several examples, Section 3, the second part of this article, shows how various kinds of information useful for risk/emergency management can be represented or categorized for the purpose of general KS. Section 3.1 illustrates how organizing and representing a small terminology, and why performing such tasks is important. Section 3.2 provides a general model for organizing and representing *Search&Rescue* information; the logic-based representation of procedures and other description objects is illustrated and is original for such tasks. Section 3.3 shows KRs for an automatic systematic exploration of a disaster area, e.g. by a rover (in this article, "rover" refers to an autonomous small vehicle such as those used for planetary surface exploration); the illustrated originality in Section 3.3 is the representation of procedures. Section 3.4 represents information about ways to design rovers that are adapted to a terrain; the illustrated originality is in showing how all the important information from three different research articles are synthesized, related and organized. The contents of all these KRs (models, procedures, techniques, …) and their use for designing the intended rovers are themselves validated by the designed prototype rover and its capabilities [17].

2. Four Complementary Avenues for Supporting 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. From a graph-oriented viewpoint, KRs are concept nodes (i.e. concept type instances, quantified or not) connected or connectable by relation nodes (or, more shortly, "relations": existentially quantified instances of relation types). KRs are expressed in formal languages: KR languages (KRLs). In this article, a KB is a set of objects that are either types (objects that can have instances) or non-type objects. Statements (KRs) are non-type objects. Types are either concept types or relation types. In this article, a "term" is an object identifier that does not solely come from the used KRL, i.e. that is not solely predefined. A term is *defined or declared* in an ontology. A KB is only an ontology if it has no base of facts, hence if all its statements are definitions. Box 1, Box 2 and Section 3 give KR examples. Box 3 illustrates simple semantic queries on KRs.

Box 1. Some equivalent formal representations of a very simple statement (in the names of the given KRLs, "/" separates the used "logic/abstract model(s)" part from the used "concrete syntax model" part, and means that the first one is linearized with the second one).

English: By definition, a flying_bird_with_2_wings is a bird that flies and has two wings.

PL (Predicate logic; here, more precisely, "First-order_logic / Modern_variant_of_the_Peano-Russel_notation"):

```
Flying_bird_with_2_wings (b) := Bird(b) ∧ ∃f Flight(f) ∧ agent(f,b) ∧
                                ∃w1,w2 Wing(w1) ∧ Wing(w2) ∧ part(b,w1) ∧ part(b,w2) ∧ w1!=w2
```

Notes: an "agent" relation links a process to its "do-er" hence, in natural language grammars, to its "subject";
in the KRs of Section 2, italics are used for relation types and only for these terms;
fully understanding these representations is here not required: they are only intended as examples.

First-order_logic / Prefixed-KIF (note: KIF represents concept types as unary relation types):"):

```
(defrelation Flying_bird_with_2_wings (?b) := (exists ((?f Flight) (?w1 Wing) (?w2 Wing))
                          (and (Bird ?b) (agent ?f ?b) (part ?b ?w1) (part ?b ?w2) (/= ?w1 ?w2))))
```

FE (Formalized-English; here, more precisely, "First-order_logic / FE_notation"):

```
any Flying_bird_with_2_wings is a Bird that is agent of a Flight and has for part 2 Wing.
```

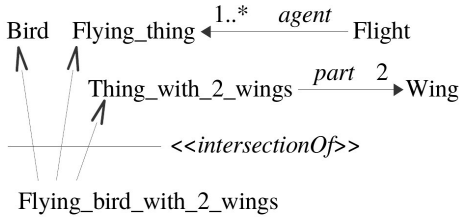
FL (here, more precisely, "First-order_logic / FL_notation"):

```
Flying_bird_with_2_wings  =  ^(Bird  agent of: a Flight,  part: 2 Wing).
```

RDF+OWL2 / Turtle (a language advocated by the W3C and commonly used for Linked Data):

```
:Flying_bird_with_2_wings  owl:intersectionOf
   (:Bird  [a owl:Restriction; owl:onProperty :agent;     owl:someValuesFrom :Flight]
           [a owl:Restriction; owl:onProperty :wingPart;  owl:qualifiedCardinality 2]).
```

UML (here, more precisely, "UML_model / UML_concise_notation"):



Legend for this graphic notation:

- each arrow "->" represents a supertype (sublassOf) link
- for other links, the arrow "→" is used with an associated link type and also a destination cardinality when this cardinality is different from 0..*, i.e. 0–N
- in the used concise notation, boxes around classes (types) and associations (links) are not drawn.

Box 2. Some equivalent formal representations of a more complex statement, one that cannot be represented in first-order logic (and, a fortiori, in RDF+OWL2; for the representation with the Turtle notation, the IKLmE logic and structural model is used).

English: On March 21st 2016, John Doe believed that in 2015 and in the USA,
at least 78% of adult healthy carinate birds were able to fly.

FE:
```
` ` ` ` `at least 78% of Adult Healthy Carinate_bird is able to be agent of: a Flight´
at place USA´ at time 2015´ for believer John_Doe´ at time 2016-03-21´.
```

FL:
```
[ [ [ [ [at least 78% of Adult Healthy Carinate_bird is able to be agent of: a Flight ]
place: USA ] time: 2015 ] believer: John_Doe ] time: 2016-03-21 ].
```

IKLmE / Turtle:
```
[rdf:value
    [rdf:value
       [rdf:value
           [rdf:value
               [rdf:value [rdf:value [:agent_of [a :Flight]
                                   ];  pm:q_ctxt [quantifier  "78to100pc";
                                                  rdf:type  :Adult, :Healthy,
                                                            :Carinate_bird ]
                          ];  pm:ctxt [:modality  :Physical_possibility]
               ];  pm:ctxt  [:place  :USA]
           ];  pm:ctxt  [:time  "2015"]
       ];  pm:ctxt  [:believer  :John_Doe]
    ];  pm:ctxt  [:time  2016-03-21] ].
```

Box 3. Some equivalent formal representations of two semantic queries on a KB.

English: In this KB, what "minimal graph" implies that some/all birds have or may have 2 wings ?
(notes: "minimal graph" here means that the shorter-but-still-correct answer, the better;
otherwise, if there is an answer, the whole KB would also be an answer;
in FL an FE, the query operator "?" is used for retrieving such graphs in a KB;
the statement represented in Box 1 is one answer to this query)

FE:
```
Is there a statement ?s that has for implication `a Bird may have for part 2 Wings' ?
```

FL:
```
?s  [?s => [a Bird part: 2 Wing] ]
```

SPARQL (a query language advocated by the W3C; since, SPARQL is not able to represent the above query, the statement represented in Box 1 is one answer to this query):
```
CONSTRUCT { ?b ?rPart ?w }  WHERE { ?b rdf:type Bird .   ?w rdf:type Wing .
                                   ?rPart rdfs:subPropertyOf* :part  .  ?b ?rPart ?w }
```

English: Which birds in this KB are described as having 2 wings ?

FE:
```
? ?b  `a Bird ?b that has for part 2 Wing'
```

FL:
```
? ?b  [a Bird ?b  part: 2 Wing]
```

SPARQL:
```
SELECT ?b  WHERE { ?b rdf:type Bird .   ?w rdf:type Wing .
                   ?rPart rdfs:subPropertyOf* :part .  ?b ?rPart ?w }
```

When it comes to KR languages (KRLs), the W3C first proposes *a few ontologies for "KRL models", i.e. logic and structural models*, e.g. RDF for representing very simple logic formulas (existentially quantified conjunctive formulas), OWL2 for the use of the SROIQ description logic and RIF for representing rules of more expressive classic logics. The W3C also proposes *some notations, i.e. concrete syntax models, for the previous KRL models*, e.g. the notations named RDF/XML, RDF/Turtle and RIF/XML. Box 1 illustrates RDF/Turtle and the meaning of "/" in these names. There exists other standards for other KR logic models, e.g. the model of KIF (the ANSI "Knowledge Interchange Format") and Common-Logic (CL, the ISO/IEC model for first-order logic), with various notations for them, e.g. Prefixed-KIF, Infix-KIF and XCL ("XML for CL"). However, as described by the next two paragraphs, the current standard or common KRLs have at least two problems for general KS, e.g. for risk/emergency management.

The first drawback of these KRLs is their expressiveness restrictions. Although these restrictions ensure that what is represented via these KRLs has some interesting properties (e.g. efficiency properties), these restrictions *prevent* the representation of some useful information: some KRs cannot be formally written. Then, these KRs cannot be shared, and this also often leads to the writing of KRs in *ad hoc*, imprecise or biased ways, hence in incorrect or far less exploitable ways. Conversely, for general KS, enabling people to write expressive KRs has often no downside since, when needed and whichever their expressiveness, KRs can be translated into less expressive ones. This can often be completed automatically, to fit the need of a particular application, by *discarding* the kind of information that *this* application cannot handle or does not require. Since such choices are application dependent, the knowledge users should make them, not the knowledge providers. KRs designed for particular applications are often unfit (too biased or restricted, ...) for other applications. As mentioned in other words within the introduction, in general KS, knowledge providers do not make application-dependent choices – or only as additional specializations, hence without restricting the possibilities of knowledge users. Since current or future risk/emergency management cannot be reduced to a list of particular applications, it is limited by expressiveness restrictions.

A second important drawback of these KRLs is that they are not "high-level", meaning that they are not supporting or leading to "normalized and easy to read or write" representations of many important notions such as numerical quantifiers, meta-statements, and interpretations of relations from collections. Hence, *even when similar pieces of information are represented*, if different KRLs or different knowledge providers are involved, the results are generally so different that matching them to each other is difficult to do automatically, and hence so is searching or aggregating them. Using *ontology design patterns* – such as those of [18] – is difficult and only very partially addresses these issues; thus, it is rarely performed. In addition, for different domains or applications, it is often useful to use different notions and different ways to represent information. Viewing – and, *a fortiori*, writing – KRs via current KR editors is even more restricting in terms of what can be displayed and expressed. E.g., graphics take a lot of space and thus do not allow people to simultaneously see and hence visually compare many KRs (this is a problematic for KR understanding and browsing).

A first answer to these problems was (i) FL [19], a KRL that has a very expressive, concise and configurable textual *notation*, and (ii) FE [20], an English-looking version of FL which can more easily be read by people with only a small training in KR. Like FL, FE can use an ontology even for logic-related terms such as quantifiers and hence can be a notation for any logic, unlike the other logic-based controlled languages, e.g. "Attempto Controlled English" and "Common Logic Controlled English". Box 1 and Box 2 illustrate the expressiveness and high-levelness of FL and FE compared to some classic KRLs. The English statement in Box 2 could have been represented in KIF (since it has a second-order logic notation interpreted into a first-order logic model) but in a less readable and normalizing way.

A more general and complementary answer is the design of an ontology of (i) *model components* for logics, *and* (ii) *notation components* for these models. KRLO (KRL ontology) [21] is a core for such an ontology: it supports the definition of KRL languages (and actually most formal languages). Furthermore, it is stored in a cooperatively-built shared KB (details in Section 2.3), that allow Web users to extend KRLO and store the definitions of new KRLs. A library of software components exploiting such an ontology is currently being created. Via these components or modules, KB systems will be able to import/export from/to/between any such specified KR languages, and thus also perform particular kinds of KR translations (in addition, since the rules for such translations are also specified in the ontology, tool users will not only be able to select the rules that they want to be applied but also complement these rules). [22] criticized KIF, and other would-be KRL interoperability standards, for necessarily packaging only a particular set of logic primitives and hence not actually supporting interoperability if the primitives of any logic cannot be defined with respect to each other with that KRL. The use of KRLO and translation-procedures-based on it is a solution to this problem and can be seen as a way to have the interoperability advantage of standards without their expressiveness and notational restrictions. [21] also shows how common notations such as Turtle or JSON-LD can be used for representing meta-statements and many kinds of quantifiers, albeit in a yet non-standard way. Box 2 illustrates this with Turtle and IKLmE, a model that is part of KRLO and that represents the concept and relation types of IKL [23], a first-order logic model that is an extension or variant of CL and KIF for

interoperability purposes. Some other research projects had or have some similarities with the KRLO project but do not share the goal of supporting one shared ontology for any number of KRLs. Furthermore, KRLO is cooperatively extendable by Web users, as detailed in subsequent subsections, for general KS purposes as well as general translation purposes between KRLs. No other project related to KRL ontologies had the same goal as the KRLO project. The LATIN (Logic Atlas and Integrator) Project (2009 – 2012) [24] represented translation relations between many particular logics. Ontohub [25] is (i) a repository that included some KRL model representations and some translation relations between them, and (ii) an inference engine able to integrate ontologies based on different logics. ODM 1.1 [26] is an ontology that relates some elements of some KRL models, mainly RDF, OWL, CL and Topic Maps.

2.2. General Purpose Ontologies Merging Top Level Ontologies and Lexical Ones

Foundational ontologies or, more generally, top-level ontologies define types that *support and guide* the checking, organization and representation of the ontologies they are included in. Two examples of well-known general foundational ontologies are DOLCE [27] and BFO [28]. The previously cited POLARISCO [8] relies on BFO for better formalizing and relating the terminologies and methods of various emergency response organizations.

Strictly speaking, lexical ontologies – e.g. ConceptNet 5.5 [29] – organize and partially define various *meanings of words from natural languages* and relate these words to these meanings. However, in this article, the expression "lexical ontologies" also refers to "large mappings between general KBs", e.g. the lexical ontology of UMBEL (now retired but included into KBpedia [30]) which had more than 35,000 types and 65,000 formal mappings between categories from (for example) OpenCyc, YAGO, DBpedia, GeoNames and schema.org.

Both kinds of ontologies – top-level ones and lexical ones – are domain-independent, thus usable in risk/emergency management. The more a KB reuse types from such ontologies, the easier it is for people to create, update or organize this KB and the more any of its content can be retrieved using these types. Similarly, the more types two KBs share and are based on (hence, especially types from such ontologies), the easier the content from these two KBs can be *aligned or fully integrated*. Below, the word "*merge*" is used for referring to any of these two processes. Since such ontologies are sets of *definitions*, as opposed to assertions of facts or beliefs, inconsistencies between these ontologies are telltales of conceptual mistakes, such as over-restrictions or misinterpretations. Thus, for the parts these ontologies are not redundant with one another, such ontologies complement each other and, possibly after some making some corrections, can be *merged* without this leading to inconsistencies.

The Multi-Source Ontology (MSO) [31] is a step towards such a merged ontology. The MSO already merges several top-level ontologies as well as a lexical ontology derived from WordNet [32]. It will be complemented with other top-level ontologies, typically those from other merges included in large general ontologies such as YAGO and DBpedia. However, unlike for other merges, the ones in the MSO follow the general KS supporting methods described in the next subsection. Here are examples of what this entails.

- The MSO is in a cooperatively-built shared KB where it can be improved and complemented by Web users.
- Modifications in such a KB are, whenever needed, "additive", as opposed to "destructive", since (i) a modification can be made by adding a relation that states how a newly entered KR *corrects* another KR, (ii) KRs are represented as *viewpoints, preferences or beliefs* from particular knowledge providers, and (iii) particular relations must be entered between opposing beliefs for them to be later automatically managed according to the wishes of each user. The next sub-section explains how. The other KS approaches are essentially based on helping the creation, handling, retrieval and aggregation of *(possibly competing) ontology modules* – e.g. see [33] – and *versions* (for KBs, hence for ontology modules too) – e.g. see [34]. Modules and versions are relation sets which may be "partially redundant and inconsistent" with each other, i.e., which may be competing. Thus, when creating a KB, such sets often require choices by ontology designers or users for selecting one or another. Using different modules or versions lead to different KBs, thus increasing the list that some knowledge users have to choose from and sometimes integrate. With the approach used in the MSO, additions do not require choices between relations and particular modules or versions can still be extracted using semantic queries.

- In accordance with the previous point, when an ontology is merged into the MSO, its content does not need to – and is not – destructively modified to fix conflicts with other ontologies. Thus, no arbitrary choice has to be made and this eases the integration of later versions of these integrated ontologies.
- The MSO has a top-level organized by subtype partitions, and thus has advantages similar to those of a decision tree for knowledge inference and retrieval purposes. This organization is kept when new KRs are added into the above-cited kind of "additive but consistent" shared KBs.

In addition to a lexical ontology and top-level ontologies, the MSO includes KRLO and hence types interesting for categorizing or representing software or procedures. Section 3.2 shows how this last point is useful for risk/emergency management too.

2.3. KB Servers That Support Non-Restricting KB Sharing by Web Users

A user of a shared KB may want to complement it with a statement that contradict another knowledge provider's statement already in this KB. However, for general KS purpose, a KB should not include two statements that are logically inconsistent with one another, since classic logics – and therefore most inference engines – cannot handle KBs that are logically inconsistent (in other words, most KB management systems are not based on a paraconsistent logical system or a similar approach). Similarly, for general KS purpose, avoiding inconsistencies in a shared KB cannot be achieved by having a person or a committee decide to *accept or not* each new statement that is submitted to the KB. Indeed, this process is too slow to be scalable and it is important for *general* KS to preserve the possibilities for knowledge end-users to make selections themselves according to their particular needs. Similarly, general KS cannot use solutions based on selecting only consensual KRs or only KRs from a largest consistent subset of the KB. Using a software to dispatch the submitted statements into different KBs (depending on various criteria) for each resulting KB to be internally consistent, e.g. as in the Co4 protocol for building consensual KBs [35], is also not a scalable solution: with such a method, the number of required KBs can grow exponentially and these KBs may be mostly redundant with one another.

Solutions start by associating *each term (alias, identifier within the KB) and statement* to its source (its author or, if unknown, the source document). This is already a standard practice when it comes to terms (alias, object identifiers), e.g., the systematic use of URLs (with or without abbreviations) is advocated by the W3C. Regarding statements, making this association is to acknowledge that the statements which are usually called *facts* in KBs are actually *beliefs*: the associations between them and their sources become the actual facts. This association may be made via meta-statements that contextualize other statements to represent who created these last ones or believe in them. (Unfortunately, as of 2022, the W3C has not yet made recommendations regarding ways to represent contextualizations and OWL does not support the representation of meta-statements). More generally, in KBs that include such *beliefs*, the statements provided by users can be categorized as being either "beliefs" or "definitions". These last ones are always "true, by definition" since the meaning of the term they define is whatever its definitions specify (thus, if a definition of a term is inconsistent, this term means "*something impossible*"). For example, assuming that `pm` is an identifier for a particular user in a KB, then `pm` is entitled to create the term "`pm:Table`" (this identifier uses the term-prefixing syntax allowed by most KRLs advocated by the W3C) and to define it as a type for flying objects rather than as a type for some kinds of furniture. Thus, *definitions* do not need to be contextualized like beliefs are.

Thus, to avoid direct inconsistencies between statements from different contributors (knowledge providers), a shared KB may have an editing protocol that leads to the entering of beliefs instead of facts. When a contributor C is about to add a belief that the inference engine detects as being *in conflict or partially redundant* with another contributor's belief already in the KB, the protocol may ask C to relate the two beliefs for (i) representing why this addition is necessary (this is also a way to make C realize that the addition is not necessary or has to be refined), and then (ii) let the inference engine exploit such relations between conflicting beliefs for making choices between them when such a choice is required. For example, if the statements "according to user X, birds fly" and "according to user Y, healthy adult carinate birds can fly", then a relation must be added between these statements to state whether the second statement is a correction (by Y) of the first statement, or whether the first statement is a correction (by X) of the second statement. Such a relation can then be exploited (according to application requirements

or the preferences of the current user) for automatically or manually selecting which statement should be exploited by the used inference engine for the cases when this engine must choose between the two statements. If the purpose is simply to retrieve knowledge, this choice may not be needed since, when two statements are potential answers to a query, a good and informative result may simply be to return both of them connected by the relevant corrective relation. One particular rule for an automatic exploitation strategy may be a specification of the following informal rule: "when a choice between conflicting statements from trustable authors is needed, select the most corrected statements according to their inter-relations and then, if conflicts remain, generate all maximal sets of non-conflicting statements and give the results of the inferences made with each set". Different users may refine or complement this rule in many ways.

The *shared KB editing protocol* of the WebKB-2 server [36] implements and actually adds some precision to this general approach. This protocol uses the addition of particular relations to the KB not only to be able to manage KB sharing conflicts but also modifications to the KB: modifications are *additive*, not destructive. For example, when objects (relations or terms) are made obsolete by their creators but are still used by other agents, these objects are not fully removed but contextualized in a way indicating (i) regarding terms, who their new owners are, and (ii) regarding relations, who do not believe in them anymore. Regarding the addition of a belief that the inference engine detects as being *in conflict or partially redundant* with already stored ones, the *main principle* of this protocol is to ask the author of the belief to *connect* it to *each of these particular other stored ones* via a relation of a type *derived* from each of the following ones: "`pm:correction`", "`pm:logical_implication`" (alias, "`=>`") and "`pm:generalization`" (not all logical implications are generalizations). Here, "derived" means either "identical", "inverse", "exclusive with", "subtype of", "subtype of the inverse", or "subtype of a type exclusive with". E.g., "`pm:non-corrective_specialization_only`" is defined as a subtype of the inverse of "`pm:generalization`" as well as an exclusion to both "`pm:correction`" and "`=>`". Thus, *all* potentially conflicting or redundant statements are (directly or transitively) connected via these relations. This organization has many advantages for inferences, quality evaluations and checks of the KB, e.g. statements can be searched for via their exclusion to some other ones. Even more importantly for general KS, this organization supports automatic choices between conflicting statements via rules such as the one given in the previous paragraph.

Since knowledge providers can specify the above-cited relations even when an inference engine is not able to detect potential conflicts or implicit redundancies, knowledge providers can also specify such relations between *informal* statements within a KB or a semantic wiki. Thus, the above-described approach can *also* be used for organizing the content of a semantic wiki and thus avoiding or solving edit wars in it. To sum up, the approach described in the previous paragraph works with any kind of information, does not arbitrarily constrain what people can store or represent, and keeps the KB organized, at least insofar as people or the used inference engine can detect redundancies or inconsistencies. In a fully formal KB, many implications have to be provided by knowledge providers (e.g., these implications may be rules these persons believe to be true) but generalization relations between statements can be automatically generated, e.g. for inference efficiency purposes. To obtain or keep a partially informal shared KB *organized*, and hence better exploit it for inferences and cooperation purposes, the more this KB uses some informal terms in its statements, the more it is useful to also ask the knowledge providers to specify generalization relations between statements.

2.4. KB Servers That Support Networked KBs

As hinted in the introduction (first paragraph), there is a huge amount of information that can be valuable for a domain such as risk/emergency management (and the information can also be used for many other purposes). All the information cannot be stored into a single *individual KB* (alias, *physical KB*). An *individual KB* is a KB having one associated *KB server* that stores this KB and manages query/update accesses to it – one server or, for security purposes, a set of equivalent ones. As opposed to such a KB, a *networked KB* (alias, *virtual KB*) is composed of a network of individual KBs where the KB servers exchange information or forward queries among themselves.

The W3C has not made recommendations about networked KBs, it only advised KB authors to relate the terms of their KBs to terms of some other KBs. This advice tries to reduce the problems coming from the fact that most KBs are developed *independently* from one another, and hence are *just structured data for one another* since their ontologies

are not related or poorly related. However, this strategy for *partially independent* development of KBs only very partially solves the above referred problems: the more knowledge is added to such KBs, (i) the more inconsistencies and implicit redundancies they have between them, i.e. together, (ii) the harder it then is to align or integrate them, and (iii) *each* user wanting to reuse such KBs *has to (re-)do* such an integration work. Although there are numerous approaches for partially automatizing such a work or aspects of it, as for example recently summarized by [37], their success rates are necessarily limited: correctly and fully integrating two (partially-)independently developped ontologies requires understanding the meaning of each object in these ontology and hence, most often, finding information that is not represented in them.

Thus, for reasons similar to those given in the previous (sub-)sections, requirements for a networked KB to be scalable and interesting *for general KS purposes* are: (i) its overall content, i.e. the sum of its component KBs, should be as organized *as if* it was stored into one individual shared KB with the properties described in the previous subsection, (ii) neither the repartition of the KRs among the KBs, nor the process of adding an 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. Thus, *ideally, for general KS on the Web*, (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.

These constraints are not satisfied by networked KBs based on *distributed or federated database systems*. Indeed, in these systems, the protocols that distribute or exchange information and forward queries exploit the fact that each individual KB or database has a *fixed* database schema or ontology, i.e. one that is not modified by its end-users (e.g. data providers). On the other hand, in general KS, the ontologies of the individual KBs are *often* updated by their contributors. Many networked KB architectures exploit such database systems, including the architectures advocated in risk/emergency management related articles, e.g. those of [15].

Similarly, these constraints are not satisfied by networked KBs based on *peer-to-peer (P2P)* protocols or *multi-agent system (MAS)* protocols. Indeed, for exploiting the KRs within these KB – e.g., for the distribution, indexation or exchange of knowledge – these protocols also have to rely on some *fixed and/or centralized* ontologies (and/or use knowledge similarity measures or automatic ontology integration techniques when these approach are sufficient for the intended task or domains; these measures or techniques may be provided by the individual servers, peers or agents). These fixed ontologies may be stored within the individual servers, software agents or peers – or sometimes even the P2P routing table, as described by [38]. They may also be external to them, with more structured networks (e.g. the use of super peers) or centralized solutions, for instance as described by [39–41].

For satisfying the above-cited constraints, the solution proposed in [19] by the first author of this present article 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 recent extension this solution by the first author. An *intensional scope* is a KR specifying the kinds of objects (terms and KRs) that a shared individual KB server is committed to accept from Web users. This scope is chosen by the owner(s) of this shared individual KB. An *intensional core scope* is the part of an intensional scope that specifies the kinds of objects that the 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 Web document that lists each formal term (of the ontology of the individual KB) that uses a normalized expression of the form "<formal-term-main-identifier>__scope <URL_of_the_KB>". Since extensional scopes are Web documents, such a format enables KB servers to exploit Google-like search engines for retrieving the addresses of KBs storing 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 within 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 the selected 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 as intensional scope. Thus, this server can be used by any networked KB as one possible nexus for non-domain specific terms

and KRs.) Then, "an individual KB (server) *joining* 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 similar or better properties.
- 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 their published extensional scopes.

Thus, thanks to this propagation, each *command is forwarded to all nexus that can handle it*, and *no KB server has to store all the terms of all the KBs*, even for interpreting the published scopes of other nexus. To counterbalance the fact that some forwardings of KRs may not be correctly performed or may be lost due to network errors, i.e. to counterbalance the fact 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". KB servers that have overlapping scopes may have overlapping content but this redundancy is not implicit and hence, as explained in the previous subsection, not harmful for general KS purposes.

To sum up, Section 2.4 showed how some inter-KB organization(s) can replicate an inner-KB organization that has advantages and supports that are described in Section 2.3 and Section 2.2, which themselves are made possible via the language-related techniques introduced in Section 2.1. Section 3 illustrates some applications of some ideas from Section 2.1 and Section 2.2 for some knowledge useful in risk/emergency management.

3. Examples of Representations for General Knowledge Sharing

In the present section, for clarity and concision purposes, the FL notation [19] is used rather than a W3C KRL notation. Thus, for identifiers, the namespace end delimiter is "#" (as in `pm:Table`) instead of ":" in W3C KRL notations (as in `pm:Table`); indeed, in FL ":" is the end delimiter for relation nodes, as in most frame-based KRLs.

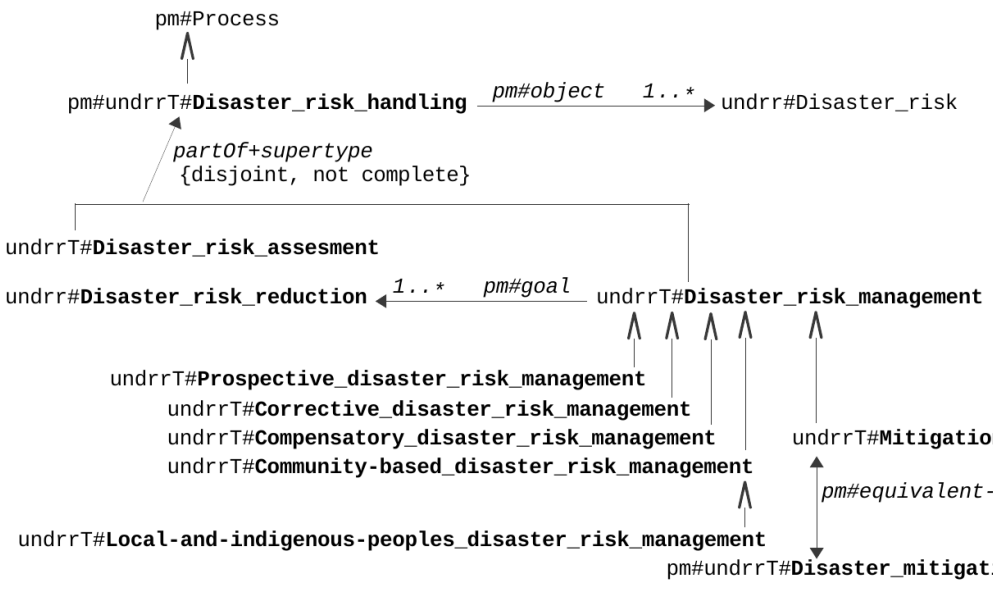
3.1. Organization of a Small Terminology about Disaster Risk Reduction

In 2017, the United Nations office for Disaster Risk Reduction (UNDRR) has defined a "terminology about disaster risk reduction [42]". It is here now referred to as "UndrrT". In [43], we represented UndrrT in FL, increased its organization and stored it in a Web document. As illustrated by Figure 1 – which uses the Uniform Modeling Language (UML) [44] – and Box 4 (which uses FL [19]), this document organizes UndrrT into a subtype hierarchy that uses (i) whenever possible, subtype partitions or other subtype exclusion sets, (ii) the MSO top-level concept types, and (iii) some additional types when this is required for categorization purposes. This Web document – which is both an HTML document and a KR storing document – is also informally structured via sections and subsections, with respect to some of the MSO types, thus in a non-subjective and systematic way. Thanks to these various points, the terms and relations between the terms in UndrrT are much easier to understand and retrieve (by following relations between them or via queries) than terms in the original UNDRR document: these last terms are only informally defined and only listed in alphabetic order.

Among three points listed in the previous paragraph, the first two also enable some automatic checking of the way the UndrrT terms are used in KRs or specialized by KRs, in order to (i) detect full or partial misinterpretation of some of these terms, and (ii) guide knowledge representation. E.g., instances of the type `undrrT#Disaster_risk_management` are defined to be usable as *source nodes* in relations that have a signature with first parameter `undrrT#Disaster_risk_management` or one of its supertypes. 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.), such relations are usable (and similarly by all people) from all instances of `undrrT#Disaster_risk_management`.

The use of the MSO for representing UndrrT also highlighted important ambiguities that are not resolved by the sometimes lengthy informal definitions associated with the terms. E.g., are the types `undrr#Vulnerability`, `undrr#Exposure` and `undrr#Resilience` meant to be specializations of what `pm#Characteristic_or_dimension_or_measure` means or of what `pm#State` (which refers to non-evolving kinds of situations) means? In our

UndrrT representation [43], we selected the first interpretation since then representing information using these types is easier than with the second interpretation. However, some other persons using UndrrT probably have interpreted and used these UndrrT terms as if they represented states. These two interpretations cannot be reconciled: they are exclusive. Thus, general KS is clearly reduced by such ambiguities.



Legend: the graphic syntax of UML – the Uniform Modeling Language [44] – is used for

1. supertype relations: these are the untyped upward relations; for the other relations, the type name is in italics;
2. 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);
3. 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 other details, see Box 4.

Figure 1. UML-like representation of the relations represented with FL in Box 4 (Box 4 shows a small part of the above-cited FL representation and extension of UndrrT) [44].

Box 4. Commented extract of the FL representation of the UNDRR terminology (as in Figure 1, this extract does not include relations for informal definitions and annotations but here there are many comments that explain 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 nor implicitly)
          )  //End of relations from undrrT#Disaster_risk_management
        }.  //End of the exclusion set and of all relations
```

3.2. A General Model for Organizing and Representing Search&Rescue Information

As opposed to other general ontologies, the MSO provides a type for "description instruments or results" (alias, "information objects", e.g. procedures, stories, languages, object-oriented classes, maps) *and* many subtypes for it, most of which are from KRLO. These types are useful for categorizing and representing many information objects that can be in risk/emergency management. Box 5 shows that the *Search&Rescue* domain requires many of these subtypes for categorizing information, e.g. for maps and procedures exploiting or enriching maps.

Box 5 shows the distinction between concrete and abstract information objects. It leads to distinguishing *concrete* maps from *abstract* ones. A concrete map, e.g. one displayed on a screen paper or on paper, is a 2D or 3D graphic representation of physical objects. On the other hand, an abstract map is a structural representation of a concrete map. Advanced *Search&Rescue* tasks imply that (i) search functions must exploit characteristics of map objects, and (ii) search agents doing terrain mapping or discovering victims or possible indices of victims must *add* objects to the map. Hence, structurally, an abstract map for such tasks *should not be a set of pixel representations* but should permit the storage, querying and update 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. These requirements do *not* mean that such maps should be directly *stored in a KB*, using relations. Indeed, using KRs would not only be an inefficient way to store and handle spatial coordinates or relationships of map objects, this would also make them difficult to exploit via classic programs, i.e. those only based on classic structures such as object-oriented classes. Therefore, such maps should remain *abstract data structures* but should be represented or implemented in *much richer structures* than those in binary formats for the 2D/3D abstract maps: raster image formats (pixel-based formats) and vector formats (graphics/geometry + texture based format, e.g. SVG and OBJ). More information can be described via the Geography Markup Language (GML) which uses a very restricted kind of KRL – GML is used by the Web Feature Service (WFS), an interface model created by the Open Geospatial Consortium (OGC) to support requests for geographical features across the web using platform-independent calls. However, GML is not for storage purposes. In any case, *ideally*, for each physical object, such a map would store a reference (e.g. an identifier or a pointer) to an information object representing this object *in a KB*, and this KB would support *semantic* queries *about* such objects. For classic queries – the structural and lexical ones – abstract data structures are

sufficient. For conceptual queries or navigation, semantic relations stored in data structures can be dynamically extracted and imported into the KB, when needed and based on the kinds of needed relations. To that end, FL and WebKB-2 have been extended to enable the reference to – and, when needed, automatic call of – "relation generators" (as we call them); they are represented in ways roughly similar to normal relations or to function calls.

Box 5. 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 italic characters are used for referring to terms that are listed in Box 6
//  while bold non-italic characters are simply for highlighting purposes.

Description_instrument-or-result-or-container //Alias Description_object
  Description_semantic-content                              //E.g. Logic_proposition
  Description_instrument-or-result                          //Alias Information_object
    Abstract_description_instrument-or-result               //Alias Information_object
      Abstract_description_instrument-or-result_wrt_the_described_thing
        Situation_abstract_description_instrument-or-result  //E.g. Principle_of_Coriolis_acceleration
          Process_abstract_description_instrument-or-result
         Control-structure_based_description_instrument-or-result    //E.g. While_loop, Abstract_procedure
             Abstract_function
           Declarative_based_abstract_description_instrument-or-result //E.g. Petri-Net
           Search_algorithm
             Graph-traversal_and_path-search_algorithm    //E.g. the A* algorithm
          State_abstract_description_instrument-or-result  //E.g. Object_oriented_class, Array
        Entity_abstract_description_instrument-or-result  //E.g. Path_description, Integer and each term in Box 6
      Abstract_description_instrument-or-result_wrt_the_used_method_or_instrument
        Non-declarative_abstract_description_instrument-or-result
        Declarative_abstract_description_instrument-or-result
          Semantic_abstract_description_instrument-or-result
            Semantic_description_instrument           //E.g. Java_semantic, Logic_semantic, Type
            Semantic_description_result               //E.g. Semantic_of_a_KB, Semantic_of_a_program
            Logic-independent_semantic_description_result        //E.g. Logical_statement
            Logic-dependent_description_instrument               //E.g. Logical_sentence
          Structural_abstract_description_instrument-or-result
            Abstract_data_type                        //E.g. Object_oriented_class, Array, Integer
            Structural_abstract_language-or-language_element     //E.g. Java_abstract_grammar
    Concrete_description_instrument-or-result
      Concrete_description_result                     //E.g. Java_concrete_function, Concrete_map
      Concrete_description_instrument                 //E.g. Java_concrete_grammar, Character
        Structural_concrete_description_instrument    //E.g. Concrete_data-structure_type
        Semantic_concrete_description_instrument      //E.g. Concrete_semantic-structure_type
  Description_container                               //E.g. File, Software, Web_server, KB_server
```

Box 6 shows a generic representation of such abstract maps that is useful for *Search&Rescue*: it is a list of semantic 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 precisely, Box 6 is a top-level ontology – hence a minimal general specification, listing or model – of functions and of the most interesting kinds of objects that these functions could exploit, among those useful for *Search&Rescue*. Before explaining the notation used in Box 6, it should be that the goal of this box is to represent three combinable important functions:

- A first one for *retrieving objects* (generally, *people*) within 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/potential victims at particular places in a map (since an often-used strategy is to first try to save the healthier and most socially valuable victims).
- A second one for *computing 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).

- A third one for *computing the best paths* (possibly given strategic rules and/or a search algorithm) *from a starting place to others* (thus, possibly an area) *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 search of victims).

In object-oriented (OO) programs, functions are often associated with some of the objects they exploit by being represented as methods of classes for these objects. This kind of association, which in KRLO is represented via a relation type of name "method", helps normalizing and organizing the code, and is now commonly supported by most common programming languages. Box 6 uses "method" relations since it is meant to be a minimal *general specification* of important primitive functions for *Search&Rescue*. The next three points comment this use.

- Box 6 uses"_{" and "}" to delimit the set of relations that define a type. These delimiters are not necessary in FL but are used here to make the specifications look more like those in common OO-like notations, UML textual notations and frame-based notations, and hence more intuitive to people that are used to those notations. (E.g. the separators "," and ";" are used in Box 6 where " " and "," would otherwise be used in classic FL.) However, despite this intended syntactic similarity with OO classes, genuine KRs are represented in Box 6, not just OO classes; indeed, genuine relations are used, not class attributes (unlike relations, attributes are *local* to classes and are not first-order entities).

Box 6. 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
//  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
           { }; //The body of this method could be written here, within these "{" and "}"
   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  }.
```

- One of the advantages of associating functions to information structures via "method" relations is that this supports the use of an intuitive *OO-like naming scheme* for the functions: in Box 6, see the "__" within method names. These names

follow the naming scheme "className__coreMethodName". With a genuine OO programming language, the part before the "__" is omitted because when a method is called on a particular object, this one (and hence, indirectly, its class) is specified just before the method, e.g. as in "objectName.coreMethodName" when the classic *dot notation* is used. With a KRL, methods are not *local* to an object or class – like relations from a object are not *local* to this object in the way its attributes are – and hence the name of the class has to be specified with the method name, e.g. via the above-cited normalizing and compact *OO-like naming scheme*.

- Another advantage of such associations is that, combined with the use of UML-like cardinalities (e.g. "1..3", "0..*") in the parameters of these methods or functions, they provide rather easy-to-use ways to generalize – or abstract away – implementation particularities, at least compared to programming languages. Indeed, with a programming language, *class* definitions are only tree structures and functions do not use cardinalities nor have successive default parameters; this generally forces a user of such languages to (i) cut a graph of relations (i.e. the model in the user's mind) 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 process of *systematically exploring* a disaster area (e.g. by a rover, to search for victims) can be represented *at a high-level* (as well as lower ones). The reuse of functions from the previous subsection is not shown. The focus here is to illustrate how (the elements of) procedures, tasks or processes can be variously organized and represented, *via KRs*. Box 7 provides an example systematic search procedure written in a procedural notation. Such procedures can often be automatically converted into pure functions (and this is the case of the one in Box 7), thus in a declarative way. Pure functions can then be represented via a KRL that handles functions, e.g. FL and KIF. With FL, or with KRLO and any KRL, procedures can also be directly represented in a state-based form. Once in a KB, functions and procedures can be organized via generalization relations and also generalized by more classic kinds of KRs, e.g. logical formulas representing rules.

Box 7. Commented procedure for a systematic search by a rover, one based on an infinite loop in which the only decision is to go ahead or not; the notation used here is common to C and Java but an FL version can be obtained by replacing each "(" by "_(".

```
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();
}
// Here are 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.
```

Figure 2 illustrates some relations (a partOf one and several subtype ones) between top-level tasks in *Search&Rescue*. Such relations can be exploited to categorize functions, e.g. to exploit and organize a library of functions useful for *Search&Rescue*, as explained in the introduction of this article. Such a library may for instance organize functions that represent different ways of performing similar processes. This library – and thus programs that reuse it – can also include a function selecting the most relevant of these different ways for a particular environmental context given as a parameter. Box 8 illustrates some further subtype relations from one of the tasks cited in Figure 2.

Box 8. FL categorization of the "Safe_path_backtracking" task or process mentioned in Figure 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)
     }.
```

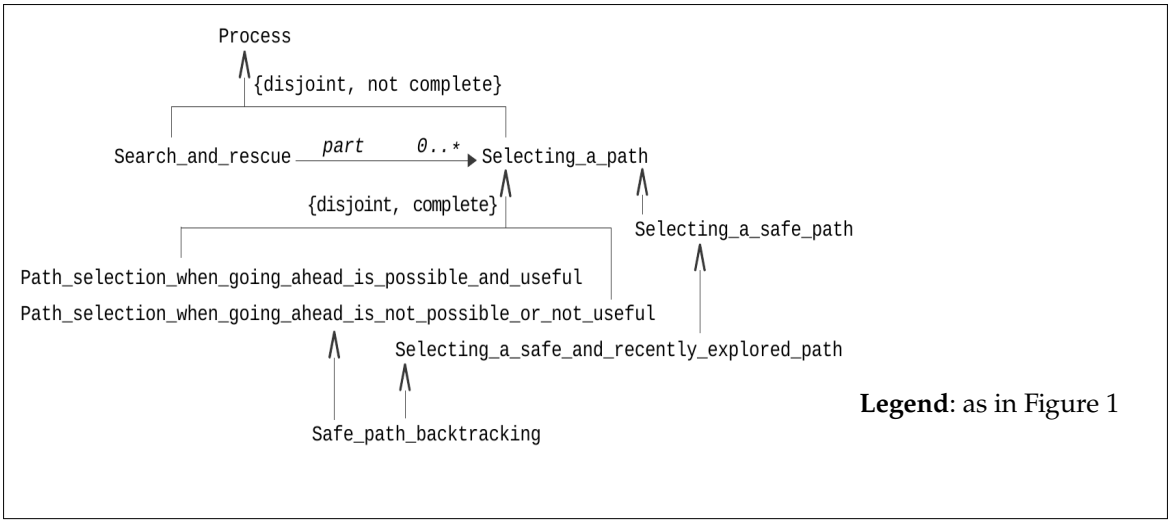


Figure 2. UML-like representationof some relations between some processes of *Search&Rescue* [44].

3.4. Representations about Ways to Create Rovers Adapted to a Terrain

The research articles of [45] or [46,47], here ordered by increasing length, describe a simulation tool helping to design rovers adapted to a terrain, for Search&Rescue purposes. Since the content of these article is in natural language, it is difficult – from this content, manually or automatically – to identify, match, represent and synthesize (i) all the important described objects (e.g., the described tasks and their instruments, subtasks, inputs, outputs, ...), and (ii) the relations between these objects. Box 9 illustrates relations from processes and software, and Box 10 illustrates relations from artifacts, attributes and descriptions. These boxes also show how the represented types are categorized as subtypes of top-level types from the MSO. The relations are representations are mostly formal but the use of informal parts (the strings within double quotes) is also illustrated since it is sometimes difficult or not worthwhile to formalize everything. Without all such relations, such objects cannot be retrieved via semantic browsing or querying. Without a shared KB (such as the ones described in Section 2.3) where such objects and relations can be found and complemented, general KS cannot be supported. Ideally, such relations should be added into shared KBs by the information authors (researchers, engineers, technicians, …). Indeed, as earlier noted, relying on knowledge engineers to read articles and represent such relations is not scalable and articles often lack the information necessary for inferring some generalization relations or other important relations.

Box 9. Commented FL representation of the "important information directly related to *processes or software*" from three articles about "3D simulation of Search&Rescue Autonomous Systems (SR-AS) and their environments for disaster management"representation of object-oriented classes for *Search&Rescue*.

```
  //From now on, "0..*" cardinalities on relation destinations are left implicit
3D_simulation_of_an_SR-AS_and_its_environment_for_a_mission  //"SR-AS": Search&Rescue Autonomous Systems
  /^ (3D_simulation  /^ Process),  //A direct supertype and an indirect one
  object: 1..* SR-AS,  //SR-AS is detailed in Box 10 (the next box)
  during: (Disaster_rescue_team_deployment time: "first hours after a disaster" ), //"...": informal representation
  part of: (Finding_a_best_design_and_configuration_of_an_SR-AS_for_a_mission
              part of: (Elaborating_a_disaster-recovery-management_strategy
                          part of: (Disaster-recovery_management /^  Process) )
           (Re-acquiring_knowledge_on_an_area_that_had_environmental_alterations
               part of: Elaborating_a_disaster-recovery-management_strategy ),
  result: (Assessment_of_an_SR-AS_for_a_mission_or_of_a_strategy_for_this_mission
             /^  Description_instrument-or-result-or-container,
             \. Best_design_and_configuration_of_an_SR-AS_for_a_mission
                Assessment_of_the_configuration_of_an_SR-AS_for_a_mission
                Assessment_of_a_disaster-recovery-mission_success_expectancy
                Assessment_of_how_much_time_or_the_SR-AS_saves_for_the_rescue_team
                Assessment_of_the_autonomous-system_survival_probability,
  input: Representation_of_the_environment_of_an_area  //cf. end of Box 10
         (Objectives_of_the_Search-and-Rescue_mission /^ State,
            part: Disaster-result_state  Disaster-victim-location_state ),
  instrument: (Software_for_simulating_an_autonomous-system_and_its_environment
                \. (3D_simulation_system /^ Description_instrument-or-result-or-container,
                      \. (Gazebo-3D  description: "graphic engine that can perform 3D physical simulations while
                                                   displacing a rover within a surrounding virtual world, hence allowing
                                                   one to test algorithms, design robots and simulate their behavior",
                           part: (Physics_engine part: Mathematical Engine, input: Scenario )
                                 Open-gestures_recognition_engine  Terrain-data_generating_engine
                                 (Graphic_rendering_engine  input: Texture  Light  Shadow) )
                   (Robot_Operating_System  description: "software usable for communication between robot parts",
                      part: (RViz description: "3D visualization environment for ROS") ) ),
  part:
    (Terrain_generation_and_modeling
       output: (Digital-Elevation-Model_of_the_terrain
                  /^ (Digital-Elevation-Model /^ Description_instrument-or-result-or-container),
                  part: (Final_Digital-Elevation-Model_of_the_terrain
                           part: First_Digital-Elevation-Model_of_the_terrain ) ),
       part: (First_phase_of_terrain_acquisition_for_terrain_generation_and_modeling
                input: (Terrain_description /^ Description_instrument-or-result-or-container,
                         \. Geographical-Information-System_data,
                         result of: (Terrain_surveying_or_mapping  \. Photogrammetry  Land_surveying,
                                      instrument:  //the next type is defined in Box 10
                                        Sensor_artefact_that_can_be_used_as_terrain_mapping_instrument) ),
                output: First_Digital-Elevation-Model_of_the_terrain )
              (Integrating_objects_and_characteristics_to_the_simulated_terrain_to_enhance_its_realism
                 input: First_Digital-Elevation-Model_of_the_terrain,
                 parameter: Physical_property,  //cf. end of Box 10
                 part: Analysis_of_properties_of_the_terrain_eg_roughness_density_bounciness_stiffness
                        Generating_random_variations_of_terrain_properties_eg_via_Monte-Carlo_distribution
                        Adding_real_obstacles_gathered_from_low-altitude-drone_flight, //rocks, rubble, …
                 output: (Final_Digital-Elevation-Model_of_the_terrain
                            description: "this includes a 3D CAD design, ground parameters (bounciness,
                                 friction, stiffness, ...), existing data, characteristics features of
                                 rigid bodies, kinematics laws, coefficients that describe an impact, …")
              ) )
    (Robot_designing input: (Robot_design_description annotation:"description of architectures/shapes (e.g.
                               via 3D CAD), physical_properties, behaviors, sensors and actuators" ) )
    (Environment_parameterization_in_the_3D_simulation
       input: Representation_of_the_environment_of_an_area ).  //cf. end of Box 10
```

Box 10. Commented representation of the "important information directly related to *artifacts, attributes and descriptions*" from three articles about "3D simulation of Search&Rescue Autonomous Systems (SR-AS) and their environments for disaster management".

```
SR-AS = Search-and-rescue_autonomous-system,  /^ Artefact,
  interest: "can reach locations unattainable or dangerous for humans",
  \. (Ground-based_SR-AS
        \. (ArcTurius_Rover annotation: "created by the LTCI laboratory of Télécom Paris",
              part: (Hokuyo_UTM-30LX_Scanning_Laser_Rangefinder_LIDAR
                       annotation: "chosen for ArcTurius_Rover because this LIDAR supports the
                                    Robot_Operating_System (ROS) communication system" ) ) ),
  part:  //there are many kinds of parts; below are examples
        (SR-AS_joint  /^ (Joint /^ Concrete_spatial-entity_playing_a_role),
           \. (SR-AS_fixed_joint annotation: "no freedom degrees")
           (SR-AS_hinge_joint annotation: "rotates along the axis and has some limited range specified by
              the lower and upper limits; can for instance be used to describe the movement of a wheel
              with respect to the chassis to which it is attached"),
          annotation: "The modeling of joints (e.g. maximum efforts+velocity they can endure) is very
                       important since (i) this permits the integration of many physical parameters, and
                       (ii) they play a key role in the physical integrity of the SR-AS after a collision" )
        Actuator_artefact_that_can_be_a_useful_part_of_an_SR-AS
        Sensor_artefact_that_can_be_a_useful_part_of_an_SR-AS.  //defined below

Sensor_artefact  /^ Artefact Sensor,
  \. (Sensor_artefact_that_can_be_a_useful_part_of_an_SR-AS
       \. (Distance_sensor_artefact \. Ultrasonic_sensor  Micro-wave_sensor  LIDAR  Camera)
          (Location-and-attitude_sensor_artefact \. Inertial-measurement-unit_based_sensor_artefact)
          Odometer_artefact  (Radar \. Ground_penetration_radar),
       annotation: "It is important to precisely model the sensors of an SR-AS (shape, size, mass,
          relative position wrt collision domain of the rover, ...), e.g. evaluating the position of
          a LIDAR for minimizing the impact of external noise. With respect to sensor modeling,
          some parameters to be taken into account while modeling a laser sensor include:
          (i) physical shape, (ii) relative poses with respect to SR-AS components, (iii) number of
          samples per unit of time, (iv) angular resolution, (v) minimum and maximum distance, and
          (vi) interference and noise (since sensors are sensitive to noise).
          For this last point, a Gaussian distribution with some moment parameterization
          (that is, given the mean and covariance of the distribution) can be used."
     ),
  \. (Sensor_artefact_that_can_be_used_as_terrain_mapping_instrument
        \. (Satellite \. TerraSAR)   Drone (Radar \. InSAR)
           (LIDAR \. Hokuyo_UTM-30LX_Scanning_Laser_Rangefinder_LIDAR) ).

Physical_property /^ Characteristic_or_dimension_or_measure,
  description: "e.g. one of the characteristics features of rigid bodies: inertia, mass, the respect of
      kinematics laws, any kind of friction, coefficients that describe the reaction to an impact, etc.".

Representation_of_the_environment_of_an_area  /^  Description_instrument-or-result-or-container,
  \. Representation_of_the_environment_of_a_disaster_area,
  description of: (Environment_situation /^ Situation,  \. Weather  Fire  Season,
                    attribute: Temperature  Humidity Magnetic_field  Pressure  Luminosity
                               (Elevation \. Depth) ).
```

4. Conclusions

The first kinds of contributions of this article were (i) its highlighting of the insufficiencies of *restricted KS* – thus, the waste of efforts and opportunities that *not using general KS* in order to support generals tasks such as risk/emergency management – and (ii) its panorama of complementary techniques that support *general KS*. Despite the fact that the problems related to traditional KR&S technologies are rather easy to be aware of when the goal is to perform general KS, this last goal is still original since (i) it requires efforts and training from knowledge providers (in exchange for less efforts and more results for knowledge consumers), (ii) developing and implementing techniques, tools and general ontologies for general KS is a difficult and very long work, and (iii) the focus of the research community is on quick and automated results since these ones are easier to publish, more granted or of more interest to the industry, and more incremental to develop and implement. Nowadays, most of general KS related research focus on the

content of foundational or lexical ontologies. Section 2.2 is about the (manual) integration of such ontologies into a unique one, something far less researched. More generally, the four subsections of Section 2 have to draw on techniques previously developed by the first author for these subsections to present techniques that are both complementary and relevant for general KS. Although some new research elements have been included, the originality of the provided panorama is in the synthesis it makes: together, the described techniques provide a rather complete approach for supporting general KS efforts useful for risk/emergency 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 particular subjects or criteria, (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? Although our framework for these points is now well developed, much more is still (and will probably always have) to be developed or implemented, e.g., more features in FL and FE, more KRLs or equivalence rules between knowledge constructs represented in KRLO, more general ontologies integrated in the MSO as well as more representations of cooperation rules within or between shared KBs – rules for the owners or users of these KBs to choose from. However, at last, these extensions can now be made by these KB owners and users.

Via Section 3, the second part of this article, the second kinds of contributions of this article were (i) KRs showing how complementary kinds of risk/emergency management related information can be represented for general KS purposes, and (ii) highlights of the interest of creating or reusing such KRs. The focused-on example domains were (i) the UNDRR terminology, (ii) a general model to represent and organize *Search&Rescue* information, (iii) tasks or procedures for automatically exploring a disaster area, and (iv) research articles about the use of a simulation tool for creating rovers adapted to a terrain. The prototype rover designed using the above represented pieces of information [17] is also validating them. Even regarding this last point, more work will have to be performed via more extensive field testing. More generally, KRs will also continue to be added to the MSO of the WebKB-2 server for supporting risk/emergency management but, since this is a huge domain, the additions will understandably be related to knowledge first needed by our own projects.

Regarding the most immediate planned extensions, WebKB-2 – and especially its procedures for evaluating or preserving the KB content quality and general KS supporting organization – will continue to be refined. These procedures allow their users to exploit the ontologies of their choices, and thus so far are generic: they have not yet proved to be domain sensitive, including in risk/emergency management.

Author Contributions: Conceptualization, P.A.M. and T.J.T.; methodology, P.A.M.; software, P.A.M.; validation, P.A.M. and T.J.T.; formal analysis, P.A.M.; investigation, P.A.M.; resources, Ph.A.M.; data curation, P.A.M.; writing—original draft preparation, P.A.M.; writing—review and editing, P.A.M. and T.J.T.; visualization, P.A.M.; supervision, P.A.M.; project administration, P.A.M. and T.J.T.; funding acquisition, P.A.M. and T.J.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data and ontologies generated or analyzed during this study are Web-accessible, e.g. http://www.webkb.org/kb/nit/o_risk/UNDRR/d_UNDRR.fl.html (accessed on 7 August 2022). In case of difficulties to find some of them, the corresponding author welcomes reasonable requests for Web addresses of these resources.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Lenat, D.; Guha, R.V. CYC: A Midterm Report. *AI Magazine*, 15 September 1990, pp. 32–59.
2. Sharma, A.; Goolsbey, K.; Schneider, D. Disambiguation for Semi-Supervised Relation Extraction of Complex Relations in Large Commonsense Knowledge Bases. In Proceedings of the 7th Conference on Advances in Cognitive Systems, Cambridge, MA, USA, 2–5 August 2019.
3. Tanon, T.P.; Pellissier, T.; Vrandečić, D.; Schaffert, S. From Freebase to Wikidata: The Great Migration. In Proceedings of the WWW' 2016, 25th International Conference on World Wide Web, Montréal, QC, Canada, 11–15 April 2016; pp. 1419–1428.
4. Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.; Hellmann, S.; Morsey, M.; van Kleef, P.; Auer, S.; et al. DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia. *Semant. Web* **2015**, *6*, 167–195.
5. Inan, D.I.; Beydoun, G.; Opper, S. Towards knowledge sharing in disaster management: An agent oriented knowledge analysis framework. In Proceeding of the Australasian Conference on Information Systems 2015, Adelaide, Australia, 30 November–4 December 2015.
6. 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 Proceedings of the ISCRAM 2008, 5th International Conference on Information Systems for Crisis Response and Management, Washington, DC, USA, 4–7 May 2008; pp. 197–207.
7. Gaur, M.; Shekarpour, S.; Gyrard, A.; Sheth, A. Empathi: An ontology for emergency managing and planning about hazard crisis. In Proceedings of the 2019 IEEE 13th International Conference on Semantic Computing (ICSC), Newport Beach, CA, USA, 30 January–1 February 2019; pp. 396–403.
8. Elmhadhbi, L.; Karray, M.H.; Archimède, B.; Otte, J.N.; Smith, B. An Ontological Approach to Enhancing Information Sharing in Disaster Response. *Information* **2021**, *12*, 432. https://doi.org/10.3390/info12100432.
9. Snaprud, M.; Radianti, J.; Svindseth, D. Better access to terminology for crisis communications. In Proceedings of the ITDRR 2016, International Conference on Information Technology in Disaster Risk Reduction, Sofia, Bulgaria, 16–18 November 2016; Springer: Cham, Switzerland, 2016; pp. 93–103.
10. Munkvold, E.B.; Opach, T.; Pilemalm, S.; Radianti, J.; Rod, J.K. Sharing Information for Common Situational Understanding in Emergency response. In Proceedings of the ECIS 2019, European Conference of Information Systems, Uppsala, Sweden, 8–14 June 2019.
11. Farquhar, A.; Fikes, R.; Rice, J. The Ontolingua Server: A tool for collaborative ontology construction. *Int. J. Hum.-Comput. Stud.* **1997**, *46*, 707-727.
12. Shadbolt, N.; Berners-Lee, T.; Hall, W. The Semantic Web Revisited. *IEEE Intell. Syst.* **2006**, *21*, 96-101. https://doi.org/10.1109/MIS.2006.62.
13. Semantic Web. W3C (World Wide Web Consortium) 2021. Available online: https://www.w3.org/standards/semanticweb/ (accessed on 8 June 2022).
14. Jain, S.; Mehla, S.; Wagner, J. Ontology-supported rule-based reasoning for emergency management. In *Web Semantics (Cutting Edge and Future Directions in Healthcare)*; Academic Press: Cambridge, MA, USA, 2021; pp. 117–128.
15. Dobrinkova, N.; Kostaridis, A.; Olunczek, A.; Heckel, M.; Vergeti, D.; Tsekeridou, S.; Seynaeve, G.; De Gaetano, A.; Finnie, T.; Efstathiou, N.; et al. Disaster Reduction Potential of IMPRESS Platform Tools. In *The Revised Selected Papers of the Proceedings of the ITDRR 2016, International Conference on Information Technology in Disaster Risk Reduction, Sofia, Bulgaria, 16–18 November 2016*; Springer: Cham, Switzerland, 2016; pp. 225–239.
16. Kontopoulos, E.; Mitzias, P.; Mossgraber, J.; Hertweck, P.; van der Schaaf, H.; Hilbring, D.; Lombardo, F.; Norbiato, D.; Ferri, M.; Karakostas, A.; et al. Ontology-based Representation of Crisis Management Procedures for Climate Events. In Proceedings of the ICMT 2018 (Workshop on Intelligent Crisis Management Technologies for Climate Events), at ISCRAM 2018, Rochester, NY, USA, 20 May 2018.
17. Tanzi, T.; Bertolino, M. Autonomous Systems for Rescue Missions: Design, Architecture and Configuration Validation. *Inf. Syst. Front.* **2021**, *23*, 1189–1202.
18. Dodds, L.; Davis, I. Linked Data Patterns—A Pattern Catalogue for Modelling, Publishing, and Consuming Linked Data. 2012, 56p. Available online: http://patterns.dataincubator.org/book/ (accessed on 8 June 2022).
19. Martin, P. Towards a Collaboratively-Built Knowledge Base of & for Scalable Knowledge Sharing and Retrieval. HDR Thesis (240p; "Habilitation to Direct Research"), University of La Réunion, France, 8 December 2009. Available online: http://www.webkb.org/doc/papers/hdr/ (accessed on 8 June 2022).
20. Martin, P. Knowledge representation in CGLF, CGIF, KIF, Frame-CG and Formalized-English. In Proceedings of the ICCS 2002, 10th International Conference on Conceptual Structures, LNAI 2393, Borovets, Bulgaria, 15–19 July 2002; pp. 77–91.
21. Martin, P.; Bénard, J. Creating and Using various Knowledge Representation Models and Notations. In Proceedings of the ECKM 2017, 18th European Conference on Knowledge Management, Barcelona, Spain, 7–8 September 2017; pp. 624–631.
22. Ginsberg, M.L. Knowledge interchange format: The KIF of death. *AI Magazine*, 15 September 1991, pp. 57–63.
23. Hayes, P. IKL Guide. 2006. Available online: https://www.ihmc.us/users/phayes/IKL/GUIDE/GUIDE.html (accessed on 8 June 2022).
24. Codescu, M.; Horozal, F.; Kohlhase, M.; Mossakowski, T.; Rabe, F. Project Abstract: Logic Atlas and Integrator (LATIN). In Proceedings of the Intelligent Computer Mathematics 2011, LNCS 6824, Bertinoro, Italy, 18–23 July 2011; pp. 287–289.
25. Codescu, M.; Kuksa, E.; Kutz, O.; Mossakowski, T.; Neuhaus, F. Ontohub: A semantic repository engine for heterogeneous ontologies. *Appl. Ontol.* **2017**, *12*, 275–298.
26. ODM: Ontology Definition Metamodel, Version 1.1. OMG Document Formal/2 September 2014. Available online: http://www.omg.org/spec/ODM/1.1/PDF/ (accessed on 8 June 2022).
27. Borgo, S.; Masolo, C. Foundational choices in DOLCE. In *Handbook on Ontologies*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 361–381.
28. Arp, R.; Smith, B.; Spear, A.D. *Building Ontologies with Basic Formal Ontology*; MIT Press: Cambridge, MA, USA, 2015; ISBN 978-0-262-52781-1.
29. Speer, R.; Chin, J.; Havasi, C. *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge*. In Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. *4444–4451*.

30. Bergman, M.K. *A Knowledge Representation Practionary: Guidelines Based on Charles Sanders Peirce*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; 464p, ISBN 978-3-319-98091-1.
31. Martin, P. The Multi-Source Ontology (MSO) of WebKB-2. 2004. Available online: http://www.webkb.org/doc/MSO.html (accessed on 8 June 2022).
32. Martin, P. Correction and Extension of WordNet 1.7. In Proceedings of the ICCS 2003, 11th International Conference on Conceptual Structures, LNAI 2746, Dresden, Germany, 21–25 July 2003; pp. 160–173.
33. Del Vescovo, C.; Horridge, M.; Parsia, B.; Sattler, U.; Schneider, T.; Zhao, H. Modular structures and atomic decomposition in ontologies. *J. Artif. Intell. Res.* **2020**, *69*, 963-1021.
34. Kauppinen, T.; Hyvönen, E. Bridging the semantic gap between ontology versions. In Proceedings of the Web Intelligence Symposium, Finnish AI Conference of 2004, Vantaa, Finland, 1–3 September 2004; Volume 2, pp. 2–3.
35. Euzenat, J. Corporate Memory through Cooperative Creation of Knowledge Bases and Hyper-Documents. In Proceedings of the KAW 1996 Canada, November 1996; Volume 36, pp. 1–18. Available online: http://ksi.cpsc.ucalgary.ca/KAW/KAW96/euzenat/euzenat96b.html (accessed on 8 June 2022). See also http://www.inrialpes.fr/exmo/papers/exmo1995.html#Euzenat1995a.
36. Martin, Ph. Collaborative knowledge sharing and editing. *Int. J. Comput. Sci. Inf. Syst.* **2011**, *6*, 14–29; ISSN 1646-3692.
37. Neutel, S.; de Boer, M.H. Towards Automatic Ontology Alignment using BERT. In Proceedings of the 2021 AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering, Palo Alto, CA, USA, 22–24 March 2021.
38. Loser, A.; Schubert, K.; Zimmer, F. Taxonomy-based routing overlays in P2P networks. In Proceedings of the IDEAS 2004, Database Engineering and Applications Symposium, Coimbra, Portugal, 7–9 July 2004; pp. 407–412.
39. Islam, M.T.; Mursalin, A.; Xuemin, S. P2P Approach for Web Services Publishing and Discovery. In *Handbook of Peer-to-Peer Networking*; Springer: Boston, MA, USA, 2010; pp. 1315–1332.
40. Gharzouli, M.; Makhlouf, D. To Implement an Open-MAS Architecture for Semantic Web Services Discovery: What Kind of P2P Protocol Do We Need? *Int. J. Agent Technol. Syst.* **2014**, *6*, 58-71.
41. Toure, M.; Guidedi, K.; Gandon, F.; Lo, M.; Guéret, C. MoRAI: Geographic and Semantic Overlay Network for Linked Data Access with Intermittent Internet Connectivity. In Proceedings of the WI-IAT 2020, IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, Melbourne, Australia, 14–17 December 2020.
42. Report of the Open-Ended Intergovernmental Expert Working Group on Indicators and Terminology Relating to Disaster Risk Reduction. UNDRR (United Nations Office for Disaster Risk Reduction). 2017. Available online: https://www.preventionweb.net/publications/view/51748 (accessed on 8 June 2022).
43. Martin, P. Representation and Organization of the UNDRR Terminology. 2020. Available online: http://www.webkb.org/kb/nit/o_risk/UNDRR/d_UNDRR.fl.html (accessed on 8 June 2022).
44. OMG Unified Modeling Language Superstructure Specification, Version 2.1.1. Document Formal/05 February 2007, Object Management Group, February 2007. Available online: http://www.omg.org/cgi-bin/doc?formal/2007-02-05 (accessed on 8 June 2022).
45. Bertolino, M.; Tanzi, T.J. Advanced Robot 3D Simulation Interface for Disaster Management. In Proceedings of the IEEE 2019 Kleinheubach Conference, Miltenberg, Germany, 23–25 September 2019; pp. 1–4.
46. Tanzi, T.J.; Bertolino, M. Towards 3D Simulation to Validate Autonomous Intervention Systems Architecture for Disaster Management. In Proceedings of the ITDRR 2019, Information Technology in Disaster Risk Reduction, Kiew, Ukraine, 9–10 October 2019; 12p, hal-02364504.
47. Tanzi, T.J.; Bertolini, M. 3D Simulation to Validate Autonomous Systems Intervention in Disaster Management Environment. In *The Revised Selected Papers of the Proceedings of the 4th IFIP Conference on Information Technology in Disaster Risk Reduction, Kiew, Ukraine, 9–10 October 2019*; Springer: Cham, Switzerland, 2019; pp. 196–211.