Ontology Design for Information Integration in Disaster Management

Grigori Babitski
SAP Research, Karlsruhe
grigori.babitski@sap.com

Florian Probst
SAP Research, Darmstadt
f.probst@sap.com

Jörg Hoffmann
SAP Research, Karlsruhe
joe.hoffmann@sap.com

Daniel Oberle
SAP Research, Karlsruhe
d.oberle@sap.com

Abstract: One of the most pressing issues in improving disaster management is that of information integration. With many organizations involved in the disaster, crossing regional or even national borders, information exchange is crucial but cumbersome due to differing vocabularies and representations both at human language and IT level. Carefully designed ontologies can be instrumental in addressing this problem, by providing a reference model for humans, and with that the basis for information integration at IT level. We devise an ontology stack covering the description of damages (caused by the disaster), resources (available to organizations fighting the disaster), and their connection (e.g. which resources are relevant for which damage). To ensure sustainable modeling, we follow the guiding principles of the foundational ontology DOLCE. We identify the most relevant design choices, and we discuss their solution. The resulting ontologies form part of a prototype that integrates information in a service-oriented architecture. We show how our ontologies are used to implement the relevant queries for information within that prototype.

1 Introduction

Disasters may be caused by flooding, earthquakes, technical malfunctions, or terrorist attacks, to name a few. The efficient handling of such emergencies, i.e., the management of the measures taken to fight them, is a key aspect of public security. This is especially true in an increasingly tightly interlinked world, where problems in one area may quickly cause problems in connected areas. This phenomenon often causes disasters to exhibit an explosive growth, especially during their early stages. Defensive measures in such a stage are still premature, leading in combination with the explosive growth to what has been termed the "chaos-phase" [Vem04]. Methods for shortening that phase are widely believed to be essential for limiting the damage caused by the disaster.

One of the characteristics of the chaos-phase is the overwhelming flow of information that must be managed by the defense organizations, such as fire brigades and the police. Many organizations are typically drawn into the event at rapid speed. To cooperate effectively, these organizations must communicate myriads of details regarding the situation, regarding the scheduling of resources, and regarding the defensive actions taken. As one might imagine, this is quite a job when organizations are heterogeneous. Fire brigades and police do not use the same technical vocabulary. What is worse, in many countries there are strong regional differences. The local fire brigade of city A might not even be able to communicate effortlessly with the local fire brigade of city B which is only 50km away.

One might say that what is needed is a standardization of terms, across all the participating organizations. A difficulty with this is that the set of "participating organizations" is not necessarily fixed – anyone may become involved if the disaster is big enough. Another problem is that standardization may be difficult for political and historical reasons; and even if human terminology is standardized, this does by no means imply that communication at IT level – e.g., between local resource databases – is possible, too. Maintaining all details within a single globalized database is often not desirable for reasons of data privacy; the maintenance is often impossible for technical or political reasons, or it simply causes too much overhead. Finally, in a world of quick technological change, standards will age fast.

In the SoKNOS project[1], we develop technology that, rather than forcing everybody to use the same terminology and IT level implementation, is aimed at facilitating the integration of heterogenous information. From the point of view of an individual organization, this means that data from heterogenous sources (where new sources may become relevant dynamically) needs to be mapped onto the organization's data schema. For such data schemata, we propose to use formal ontologies. The motivation is threefold:

1. Reasoning over the ontologies provides **superior capabilities in querying for information** (as opposed to a standard SQL-based application).

2. Ontologies **carry more information** than non-semantic schema languages such as XSD. Hence an ontological data schema is less ambiguous, which is of vital importance for the ease of schema mapping.

3. Ontologies are **better suited to serve as reference models**. Of course, different organizations will need different data schemata (indeed some of the data sources will typically be the data bases of partner organizations). We suggest to use for those schemata not a common standard, but a common *reference model*: a core ontology which each organization may *specialize* to obtain its own data schema. This ensures a large conceptual overlap, and hence greatly facilitates data integration, without enforcing identical vocabularies.[2]

Herein, we illustrate our concept with a number of use cases – queries for information – implemented in the SoKNOS prototype. Our main technical focus is on *the design of the reference ontology*. Such design is challenging since absolute clarity is a *must* – semantic ambiguities, let alone conceptual inconsistencies, are likely to cause misunderstandings. To ensure sustainable modeling, we follow the guiding principles of the DOLCE foundational ontology [MGOS03, GGMO03]. Our core contribution is to clarify how those principles are best applied in the domain of disaster management.

[1]Service-oriented architectures supporting networks in the context of public security; http://www.soknos.de

[2]Data sources may of course be integrated even if they do not use the reference model, i.e., in terms of the SoKNOS architecture that model is a suggestion, not a requirement.

Our ontology devises a high-level "core domain ontology" covering central concepts; we devise a categorization of damages (caused by the disaster), as well as a categorization of resources (the equipment available to organizations fighting the disaster). We examine how to connect the two. For example, intricacies arise in the modeling of tactical units (crews with equipment) and in appropriately modeling which kinds of resources are relevant for which kinds of damages. We discuss the issues in some detail, and motivate our solutions.

Some of the SoKNOS partners are fire brigades (Cologne and Berlin), and many of our design decisions have been taken as a result from information provided by these partners, in the form of documentation or guidelines, and in the form of feedback at workshops. Naturally, being designed for German organizations – fire brigades, police – to work with, our ontologies are formulated in German. For international use, English translations are annotated. In the presentation herein, we use those English translations.

We proceed as follows. We motivate our work with a sketch of our use cases in Section 2. We explain our basic ontology design – regarding ontology languages and the DOLCE ontology – in Section 3. This is followed by a detailed explanation of our domain-specific ontologies in Section 4. Section 5 briefly specifies how our use cases are implemented based on the ontology. Section 6 discusses related work and Section 7 concludes.

2 Motivation and Use Cases

We need to integrate information from heterogeneous organizations. One crucial example is resources availability – which organization has which resources available, and where? In current disaster management practice, such queries have to be made laboriously via telephone, constituting one of the main bottlenecks of the information flow. Clearly, this can be improved by semantic data integration [AG06], where instance data about concrete resources is fed into a domain ontology. The user can then obtain and process information via queries on the ontology. The following are a number of use cases we wish to support:

1. **Detailed browsing.** The simplest way of accessing instance information is to browse through the ontology. Each time a concept is opened, the system should return all instances of the concept, along with their known attributes.

2. **Abstract browsing.** Typically, the crisis team suffers from a huge information overload. It is hence essential to provide ways of aggregating information. One natural way to do so is via the hierarchies provided by an ontology stack. As a simple example, we can count the number of resources of a certain type, per organization. We can then show information at different abstraction levels simply by considering different levels of the hierarchy – e.g. we can count all vehicles (of any kind), or we can count all rescue helicopters equipped with intensive care equipment.

3. **Showing resource types relevant for fighting damages.** It is customary in disaster management information systems to browse for resources based on damages, showing the types of resources relevant for the respective damage. We need to cater for this by modeling the relevant connections between damages and resources.

4. **Showing concrete resources relevant for fighting damages.** We of course should also be able to show concrete resources available to fight a damage, rather than only resource types. Again, we should be able to do this at different levels of detail.

Note that abstract browsing – presentation of data at different levels of abstraction – relies on the taxonomical structure of an ontology and would, to say the least, be awkward to realize based on standard database technology.

Technically, the above use cases can be supported based on any reasonably designed ontology with the appropriate scope. However, there is a lot to be gained or lost depending on how well the ontology is designed. The two key issues are:

(I) How easy is it to integrate information, i.e., to align external terminology and data schemata with the ontology?

(II) How easy is it to systematically browse the ontology, i.e., find the category one is looking for?

Both activities are carried out by humans, and for the approach to be useful in practice both activities should be doable with ease. To what extent this is possible depends crucially on the clarity of ontology design. Regarding (II), we all know that a bad categorization makes it impossible to find things. As for (I), information integration may of course still be quite challenging even if the ontology (the target schema) is perfectly clear to the person doing the integration. However, lack of such clarity is certainly a problem. Ambiguous or flawed design will make it difficult to correctly align a data schema, and will probably lead to alignments not meeting the ontology author's original intention.

3 Basic Ontology Design

To review our basic design choices, we start with the choice of the ontology language. We then give a brief introduction to the DOLCE ontology and one of its modules.

3.1 Ontology Language

Description logic based languages are strong for conceptual modeling [BB03]. In our case, cardinality constraints, modeling primitives for a relation hierarchy, and local range restrictions are important for capturing the intended meaning of terms. In addition, we build on the DOLCE foundational ontology, whose lightweight version is maintained in description logic. Finally, the description logic based Web Ontology Language [MvH04] allows us to be standards compliant. On the other hand, logic programming approaches, such as F-Logic [KLW95], are already used successfully in commercial products (cf. [AG06]) for semantic data integration. Logic programming rules are applied to map between the heterogeneous sources and a target ontology. Further, logic programming typically allows for efficient and flexible instance retrieval, i.e., querying, which is required for our use cases.

We solve this dilemma as follows: At design time, we model our ontology in OWL DL, leveraging its expressiveness to capture our domain as precisely as possible. At run time, we transform the OWL DL ontology into F-Logic. The transformation is manually triggered once a stable version of the OWL DL ontology is achieved. We apply the transformation algorithm of the NeOn toolkit [NeO], reducing expressiveness to the class hierarchy, relations, and range restrictions. That information suffices for our run time use cases.

3.2 Foundational Ontologies and DOLCE

Foundational ontologies are serve as the conceptual foundation for domain ontologies. The latter are designed to account for a specific domain of discourse, reflecting the conceptualization of a certain user community. The two kinds of ontologies can be used in a layered architecture where meanings of symbols are inherited from the foundational level to the domain level. In this sense, the foundational ontology serves to establish the basic building blocks of the domain ontologies. This way, general errors in ontology engineering can be avoided, since basic distinctions are easier to clarify at a higher level [Mil04].

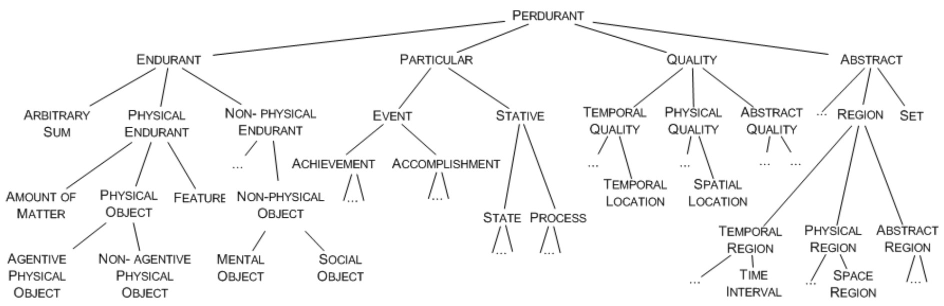

Abbildung 1: Basic categories in DOLCE [MGOS03] (slightly modified for presentation purposes).

The following provides a short introduction to the basic categories of DOLCE:

Endurants and Perdurants. Endurants and perdurants belong to the four top categories of DOLCE. They are distinct regarding their behavior in time. Endurants are wholly present at any time they exist. Their parts move with them in time. Perdurants, on the contrary, extend in time by accumulating different temporal parts. At any given time, they are only partially present, in the sense that some of their parts are not present anymore or are not yet present [GGMO03]. Perdurants embrace entities generally classified as events, processes, and activities. An endurant "lives" in time by participating in some perdurant(s). For example, a building (endurant) participates in its lifespan (perdurant). Central for our ontology of resources is the DOLCE category NON-AGENTIVE PHYSICAL OBJECT. It is a sub-category of PHYSICAL OBJECT. In contrast to agentive physical objects, non-agentive objects do not have intentions, beliefs or desires. They are what one would generally refer to as "object" in physical reality.

Qualities and Abstracts. Qualities are seen as the basic entities we can perceive or measure, for example the volume of a lake, the color of a rose, or the length of a street [MGOS03]. The main characteristics of qualities are that they are observable and that they are inherent in other entities. A quality is understood as being an individual entity itself. DOLCE makes a strict distinction between a quality, e.g., the depth of a specific lake, and its magnitude, which might be approximated via a measurement result (e.g. 10m). The "value" of a quality is considered an abstract entity.

3.3 Descriptions and Situations

Several additional theories exist for DOLCE that come in the form of ontology modules. Descriptions and Situations (DnS) is such a module and can be considered an *ontology design pattern* for (re)structuring application ontologies that require contextualization. The following paragraph provides a brief introduction, since we use DnS in our ontologies. For a more detailed description please see [GM03].

DnS introduces a distinction between *descriptive* and *ground entities*. The ground entities are categorized in the basic categories of DOLCE. The categories PARAMETER, ROLE and COURSE OF EVENT capture descriptive entities. These serve to describe the ground entities in the following way: Parameters have values, roles are played by endurants and courses of events sequence perdurants. The descriptive entities are the components of a situation description that represents the context. For example, DnS can be used to describe tactical units defining roles such as crew transporter that can be played by a bus, a van or a special crew carrying vehicle. Also of interest for our ontologies is the category DESCRIPTION since such entities underly the communication about real "world entities". A description is a communicable social object which represents a conceptualization [GM03]. For example, PLAN is a sub-category of DESCRIPTION.

4 Domain-Specific Ontology Design

We now describe our domain-specific ontologies. An exhaustive presentation would of course not be helpful; we focus on the most relevant design decisions. Figure 2 provides an overview. The following sub-sections review the core domain ontology for emergency management, the resources ontology, the damages ontology, and the ontology of deployment regulations (connecting damages to resources).

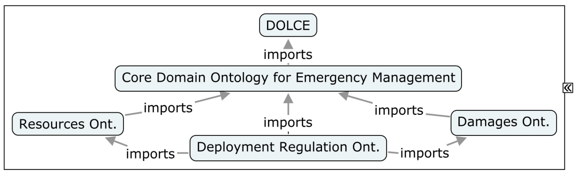

Abbildung 2: An overview of our ontology stack.

4.1 Core Domain Ontology for Emergency Management

The Core Domain Ontology for Emergency Management (short: Core Domain Ontology) forms the root of our domain-specific ontology tree, giving the basic categories and their relations. In that way, beside its role in the ontology tree, the core domain ontology also clarified terminology within the SoKNOS project. Indeed, the ontology was used to generate our common Java data structures; we get back to this in Section 5.

To clarify the basic terminology, one needs to distinguish a number of closely related categories. For example: DAMAGE (to a physical object); DANGER (occurrence prior to a damage, e.g. danger that a dam breaks); PROBLEM (the super-category of DAMAGE and DANGER; see Section 4.3); and INCIDENT (geographical and logical group of problems, damaged objects, measures, and involved resources). Further, one needs to distinguish between measures as planned by the crisis team and measures as conducted on-site; this is a good example for the importance of a common understanding (a reference model), and how alignment with a foundational ontology may further such understanding. While PLANNED MEASURE may still be ambiguous, defining it as a sub-category of DESCRIPTION clarifies for sure that a planned measure does not change the on-site situation, but instead must be realized through on-site measures. Another useful feature of DOLCE is the explicit distinction between a quality and its value. This is useful for the representation of metrics. For example, the length of a ladder is sometimes measured in meters, sometimes in terms of highest reachable floor; representing such metrics in the model is a prerequisite for converting them in data integration.

4.2 Ontology of Resources

The main task of our resources ontology is to provide a categorization which will be accessed by different organizations (fire brigades, police, etc.) during a disaster. “Resource” here encompasses a broad range of things, from sand sacks over motorized equipment (chain saws, fire engines, etc.) to specially trained personnel, and to tactical units. A comprehensive and intuitive categorization of such a broad range of things is quite a challenge; alignment with DOLCE involves a number of subtle issues as well. There is a huge gap between the fundamental categories of DOLCE and (some) very concrete notions of resources. The terminology used by fire brigades is often ambiguous and not well organized.

An example for the gap between DOLCE and the concrete resources dealt with in emergency management can be shown with the DOLCE category NON-AGENTIVE-PHYSICAL-OBJECT. In our ontology, this has been sub-categorized into DEVICE (apparat operated by humans, e.g. a helicopter, a fire engine), MATERIAL (things consisting of non-functional parts, e.g. a pipe, a barrier), ACCESSOIRES (objects that can only be used in conjunction with other objects, e.g. a battery), CLOTHING, PHARMACEUTICAL PRODUCT, and WEAPON (similar to DEVICE but with the purpose of causing damage). Obviously, from these categories several further hierarchy levels are required to reach concrete resources such as the category HAMMER. More generally, an issue is which criterion to use for the categorization. Since the ontology is to be used as a reference model, and since this is the way fire brigade men tend to think about their equipment, we have decided to use as criterion the object's effect when used according to its purpose. Purpose is essential insofar as, for example, a hammer may be used as a weapon but that is not what fire brigades will typically use it for.

The central category in the domain of resources for disaster management is TACTICAL UNIT (for brevity also referred to as TUnit). A tactical unit comprises a crew and its operating resources. For example, a fire brigade platoon is a tactical unit comprising the vehicles and men required to deal with a burning apartment. These are the organizational units that the crisis team deals with. In that sense, from the point of view of data integration, tactical units are located at the borderline between the world of particular items (a hammer, a fire engine) and the world of disaster management in the crisis team.

One important aspect in modeling tactical units is a mismatch between "theory and practice". There are detailed regulations regarding the composition of tactical units, e.g., which types of vehicles an X-Y-Z platoon of a local fire brigade consists of, which qualifications the crew should have, and what additional equipment (clothing etc.) should be present. These informations are used by the crisis team, e.g., when searching for resources. In particular, when searching for external resources (resources from organizations other than the own one), this kind of information is useful, since it provides a good impression what the external units could be used for.

The real tactical units often differ from the relevant regulations in terms of the size of the crew, and details regarding the equipment. Such details are sometimes negligible; regulations (in particular German ones) tend to be meticulous. In most cases, keeping all the details up to date in the IT system (the local database) would be impossible anyway. Usually, the only data kept up to date is which tactical unit is currently performing which operation with a crew of how many people.

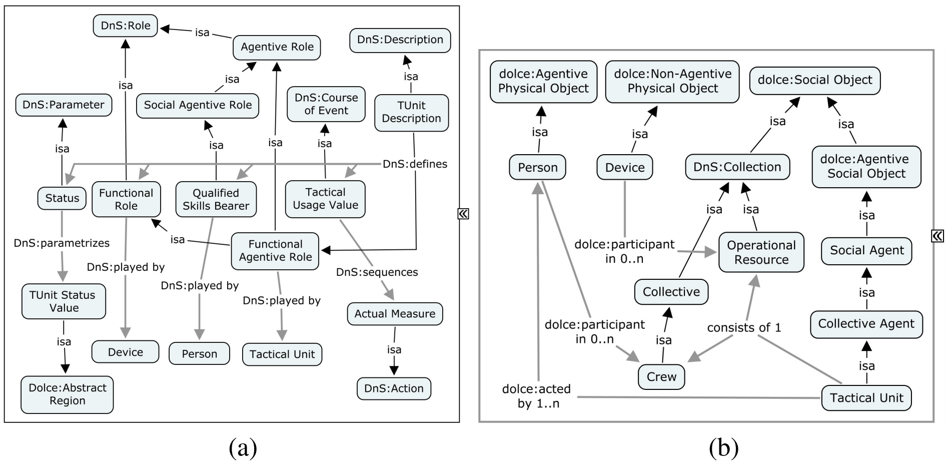

Abbildung 3: Model of (a) tactical unit descriptions (as per regulations) and (b) tactical units (as maintained in practice).

To cater for the two kinds of information about tactical units – regulations and practice – we have devised two separate models. The *model of regulations* serves as a reference, the *model of practice* serves for data integration. Both are depicted in Figure 3.

For the regulations, Figure 3 (a), we introduce to the DOLCE (DnS) category DESCRIPTION the sub-category TUNIT DESCRIPTION. A TUnit Description may define functional roles (e.g. a transporter, a tool), qualified skill bearers (a specially trained person, e.g. a group leader, a voice radio operator), and functional-agentive roles. The latter are played by TUnits. The generic "DnS:defines" relation that applies between a description and some (descriptive) entity takes the intuitive meaning of "the particular TUnit consists of other, smaller-sized TUnits". Further, a TUnit Description defines the so-called tactical usage value, i.e., what the unit can be used for.

For practice, we designed a suitable alignment with DOLCE; see Figure 3 (b). The model is minimalistic. This is appropriate because, as mentioned, the up to date information maintained about tactical units is minimalistic, too. Tactical units consist of operational resources (a DnS:collection in which devices participate) and a crew (a DnS:collective). The currently performed operation is modeled with a relation between TACTICAL UNIT (Figure 3 (b)) and TUNIT STATUS VALUE (Figure 3 (a)); Figure 3 omits this relation for the sake of readability. Similarly, TACTICAL UNIT indicates corresponding descriptions, via a relation to TUNIT DESCRIPTION, which is not shown.

4.3 Ontology of Damages

We first explain the design pattern underlying our model of damages; then we explain how the relevant entities are categorized. Figure 4 illustrates the design pattern.

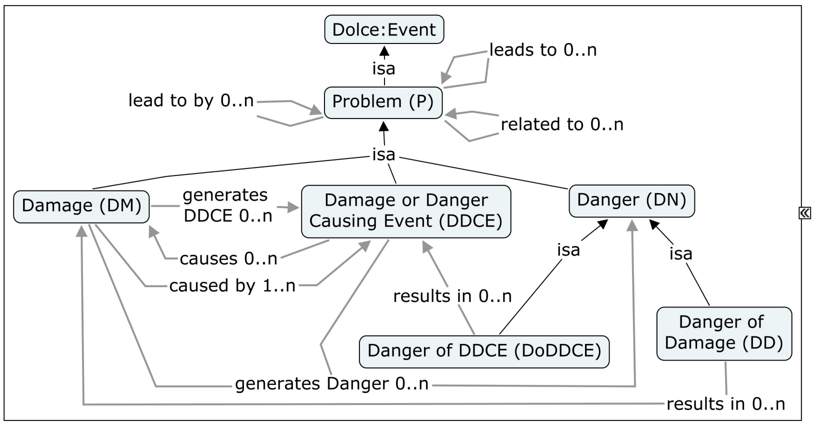

Abbildung 4: Design pattern for the description of damages.

On the top of Figure 4 we have the category PROBLEM. A problem is an event (and thus a perdurant) that may necessitate the deployment of resources. The categories DANGER (DN), and DAMAGE OR DANGER CAUSING EVENT (DDCE), comprise problems that are themselves not damages, but that may *cause* damages. Note that it is important for the crisis team to make the distinction between an already present damage and a situation that could lead to a damage. For example, a regular high tide can become a damage causing event (a flooding) if it is unusually high. Clearly, we need to be able to represent the two things (the damage and its cause) separately.

The intended meaning of the relations is as follows. The relation "lead to" is a sub-relation of "related to". Our interpretation is that there is either a causal or a temporal relation (or both) between the two problems; "lead to by" is the inverse of "lead to". Accordingly, "causes", "results in", "generates DDCE", and "generates Danger" are sub-relations of "leads to", while "caused by" is a sub-relation of "lead to by". The latter sub-relations are intended to be direct, i.e. they connect problems that lead to other problems immediately, rather than through any intermediated problems.

To illustrate the design pattern, suppose that the crisis team is notified of an explosion in a chemical factory. This is represented as the fact i_1 ∈DDCE where i_1 is a new instance and DDCE is the acronym of DAMAGE OR DANGER CAUSING EVENT as given in Figure 4. Note that the explosion as such is not necessarily a damage; but it may cause damages. For example, we may get a damage of the factory building (i_2 ∈DM) and a leakage of the gas container (i_3 ∈DM). Since those damages are caused by the explosion, we further state that causes(i_1, i_2) and causes(i_1, i_3). Due to the damaged building, there is a danger of collapse (i_4 ∈DD and generates-danger(i_2, i_4)). It is then verified that, due to the leakage in the gas container (i_3), a dangerous substance has been emitted (i_5 ∈DDCE, generates-DDCE(i_3, i_5)). The latter causes an air pollution (i_6 ∈DM, causes(i_5, i_6)) and generates a risk of further explosion (i_7 ∈DoDDCE, generates-danger(i_5, i_7)). Note in this example how problems lead to other problems, resulting in chains of problems that are causally or otherwise related. Such chains are essential in disaster management, where, proverbially, one thing leads to the other.

Instantiating our design pattern, we created comprehensive taxonomies of damages and damage/danger causing events. For the necessary classification, four criteria are conceivable: (a) What type of object is damaged (e.g. vehicle, building)?; (b) Which particular aspect of the object is damaged (its function or its structure)? (c) In which environment does the damage take place (e.g. industrial area, tunnel)? (d) What is the cause of the damage (e.g. fire, high tide)? Since the taxonomies are to serve as a reference, we need to ensure that humans can easily find a kind of problem they are looking for (by top-down browsing of the taxonomy), even if they are not familiar with the ontology. For that purpose, it is of essential importance that the categorization is consistent with respect to the criteria. Only one criterion should serve for the "top-level" categorization. Generalizing this, we use prioritized subsets of criteria. For damage/danger causing events, only criteria (c) and (d) are suitable, since a DDCE is in general not bound to a particular damaged object. We take (d) to be the primary criterion for classification; if two categories cannot sensibly be distinguished based on (d), they get categorized according to (c). For example, train-accident-in-tunnel and train-accident are foremost train accidents, and are then further distinguished by their surroundings. The reason for preferring (d) over (c) is that, when browsing the ontology, a member of the crisis team is more likely to know the type of the event than the precise surroundings.

For classifying damages, all four criteria (a) – (d) are applicable. In order to not replicate the DDCE taxonomy, we take (c) and (d) to be inferior to (a) and (b). Since the type of affected object appears to be more prominent in common perception, (a) is taken as the primary classification criterion; (b) is secondary.

4.4 Connecting Damages to Resources

Clearly, one should be able to connect damages with resources. Based on the ontology, we should be able to answer the questions: Which kind of resource is relevant to address which kind of damage? And, given some damage, which resources (actual instances) are available to address it? The first question is, for example, important if the user wishes to browse resources based on which damage they are relevant for, rather than based on their own categorization. Such browsing is common practice, e.g., in German fire brigades. The second question is, of course, essential for quickly finding relevant resources – which is one of the key goals of data integration.

How to model the connection between damages and resources? Spontaneously, one may think that a relation “requires” between problems and resources (e.g. tactical units) will do. However, matters are not that simple: different organizations use different tactical units to address the same kind of damage. Catering for this involves a number of subtle design decisions. Our design pattern is depicted in Figure 5.

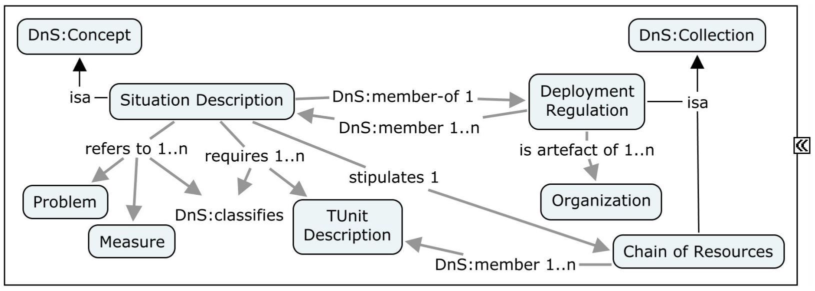

Abbildung 5: Design pattern for connecting damages to resources. (An arrow from relation A to relation B means that A is a sub-relation of B.)

In disaster management, damages are associated with resources via deployment regulations (short DR). Such a regulation is a list of situation descriptions (short SD), of which each has a name implicitly referring to a type of problem (e.g. mass injury), and a chain of resources (e.g. three rescue helicopters) relevant to address that problem. In general, each organization has its own deployment regulation; in Germany, this is true for every local fire brigade. The DRs differ in the used names and the way resources are aggregated to resources chains; they also differ in terms of granularity (e.g. some do, and some do not, distinguish between fire and explosion within a building).

Clearly, for the ontology to serve as a reference across organizations, we should be able to reflect the differences between the DRs of individual organizations. A direct “requires” relation is not suitable for this purpose, because it leaves the DR implicit. This can in principle be fixed by using the names of individual DRs as indices of concepts and relations in the ontology. But this is a contrived and inflexible model; for the sake of brevity, we omit a detailed discussion of this point.

A clean model needs to include DRs explicitly as elements of the ontology. Doing so in first-order logic involves the following difficulty. A concrete SD, e.g., "mass injury" should be modeled as an instance (it cannot be further instantiated). However, that instance does not refer to concrete resources or problems; it refers to classes of such entities. References between instances and classes necessitate higher-order logic. Assuming we wish to avoid that, there are two choices. First, "ABox modeling": a reference to a class A is represented in terms of an artificial instance a of A; e.g. we would introduce a "generic rescue helicopter". Clearly, such instances must not show up when querying the data, which at least leads to implementation and maintenance difficulties. The second option is "TBox modeling": we model concrete SDs as concepts. The problem then disappears, at the reasonable prize of a slight imprecision in what a SD is. As Figure 5 shows, at concept level we can connect SDs to the relevant classes of problems, measures, and TUnits in a straightforward fashion.

5 Prototype

The implemented ontologies serve for information integration in the prototype developed in the SoKNOS project (the prototype covers also other areas, e.g., a workflow system for planning counter-measures). We use a service-oriented architecture where legacy information stores (e.g., existing resource databases containing dynamically updated data on the status of particular resources) are encapsulated as Web services with WSDL interfaces, allowing to query/extract the contained data. For data integration, the data schemata of these Web services are aligned with our ontology. Following [AG06], we use the OntoStudio [Ont] "WSDL Web service Import" functionality; the alignment is made through OntoStudio "Mappings", a particular form of F-Logic rules editable in a GUI; and the data integration is performed, i.e., the mappings are executed, by OntoBroker [FDES98] when a query for information is made. Queries can be made in F-Logic. Concretely, we implemented the use cases outlined in Section 2:

1. **Detailed browsing.** Each time a concept **C** is opened, the system poses the query `∀X,Y,Z X:`**C**` AND X[Y->>Z]`, returning all instances `X` of **C** along with all attributes `Y` whose value `Z` is known.[3] For effective browsing, a clear use of classification criteria is essential, c.f. our discussions in Section 4.

2. **Abstract browsing.** This is accomplished with the query `∀X,Y,Z X:Organization AND Y:`**C**`[belongs-to->>X] AND count(X,Y,Z)`, where **C** is the resource type in question and *count*(., ., .) is an F-Logic built-in

[3]Note that **C** in the F-Logic query is a placeholder that will be replaced with a string – the identifier of the concept – before the query is submitted. We indicate placeholders in boldface.

aggregating the number of pairs (1st arg, 2nd arg) in its 3rd argument. Depending on the choice of **C**, we obtain information at different levels of abstraction, hence exploiting the taxonomical structure of the ontology.

3. **Showing resource types relevant for fighting damages.** We do so with the query `∀X,Y Y[refers-to=>>D] AND Y[requires=>>X]`, which will output all situation descriptions (entries of deployment regulations) `Y` and tactical units `X` relevant to the queried damage type **D**. This is a query at concept-level, which would be awkward to realize based on standard database technology, where we would need to encode the concepts and their relations as instance data.

4. **Showing concrete resources relevant for fighting damages.** We find all instances and known attributes via `∀X,Y,Z,U,V Y[refers-to=>>D] AND Y[requires=>>X] AND Z:X AND Z[U->>V]`. Here, the flexibility of F-Logic allows us to mix queries on concept and instance level.[4]

Note that all these use cases require that data sources have been connected in the first place – facilitating which is the responsibility of our thoroughly designed reference ontology.

Our ontologies also serve for integration purposes regarding the actual implementation of the prototype. The Core Domain Ontology (Section 4.1) was used to generate the common Java data structures. We first exported our OWL ontologies into XML Schema (XSD), keeping only the tree structure of the subsumption hierarchy, any cardinality restrictions, as well as the domains and ranges of the properties (aka. relations and attributes). From XSD, we generated Java data structures by a straightforward ant script; these data structures served as the basis for further development. The advantages of this approach were twofold: the connection to the ontology helped to maintain a consistent view of the domain; and the alignment with DOLCE was instrumental for reaching a common understanding in the first place.

By keeping a direct relation of Java classes to categories in the domain ontology, the approach also enables exciting opportunities for assessing the interoperability between the prototype's GUIs during runtime (a paper on this is currently under review). Additionally to the semantic annotation of the communicated information objects, the GUIs' functionalities are ontologically characterized. For example, if an object is dragged from one GUI to another, then the receiving GUI checks via the ontologies whether that object belongs to a (sub-type) of the object types it can handle.

[4]We remark that, in this use case, the deductive facilities of F-Logic – via the definition of rules – can become relevant. Particular equipment can be handled only by particular persons. It is often awkward, and sometimes impossible, to maintain this relation explicitly; an example is the measuring of particular toxins, where the measurement devices and the persons handling them may need to be acquired from different organizations. The relation can be elegantly constructed deductively by a rule referring the the required/obtained qualifications.

6 Related Work

Several projects are concerned with improving the efficiency of crisis management. One of those, AKTiveSA, also pursues an ontology-based approach to information integration [SRS$^+$07]. The developed ontology covers, amongst other things, transportation, humanitarian aid, military, equipment, organizations, and weapons.

The AKTiveSA ontology is not aligned with a foundational ontology. It is unclear whether the ontology separates between endurants and perdurants, and which assumptions are made regarding the spatio-temporal location of the two. The concept “Phenomenon” is annotated with “observable event”, but is not a sub-concept of “Event”. “Event” is annotated as “something that happens in time, i.e., it has a temporal location”; at the same time, AKTiveSA has axioms for “Phenomenon” – but not for “Event” – postulating the existence of a spatial and temporal location.

AKTiveSA introduces an “Agent” as “anything responsible for the performance of an action or participating in an activity”. According to this definition, organizations are agents, which is correct; but, e.g., vehicles are agents, too. The definition ignores the essential property of having an intent. In the terminology of OntoClean [GW02], it ignores that anti-rigid properties (e.g. agent) cannot subsume rigid properties (e.g. vehicle).

Finally, AKTiveSA makes inconsequent use of criteria for categorization. For example, the ontology contains the top-level concepts “Military related Thing” and “Agent related Thing”. “Agent” is a direct sub-concept of “Agent related Thing”, and is further subcategorized into “Organism”, “Group”, “Software Agent”, and “Platform”. So, while the top-level concepts are categorized according to something like “business area” (military, agents), this is not true of their sub-concepts which are categorized more along the lines of “physical form” (organism, software agent).

Apart from ontological models of the domain, there are some efforts to standardize some of its vocabulary. Probably the most successful one is the National Incident Management System (NIMS) used in the U.S. to coordinate incident management among various federal, state, and local agencies [NIM]. Generally, our ontologies are NIMS-compliant – they do not violate any of the underlying assumptions, and they are designed to cover much of the relevant vocabulary. We do not, however, follow NIMS closely. This would hardly be possible since our ontology is for German, not U.S., disaster management. Also, NIMS uses a rather simplified data model, covering a broad range of resources with one common table structure, which (necessarily) results in imprecisions and ambiguities.

7 Conclusion

We have designed an ontology stack covering the basic concepts of disaster management, as well as detailed categorizations of damages and resources, and the design needed for connecting the latter in an appropriate way. Our ontologies are aligned with the DOLCE foundational ontology, and have proven to be suitable for information integration in the SoKNOS prototype.

We are currently working on some final touches to the ontologies and the prototype, and we are in the process of running evaluations through disaster management practitioners. Apart from this endeavor, we are currently extending the ontology with a module for describing the data delivered by Geospatial Web services, specifically services conforming to the "Sensor Observation Service" specification of the Open Geospatial Consortium. That ontology module will form the basis of facilities for convenient discovery of such services, as well as data extraction and integration.

Literatur

[AG06] J. Angele und M. Gesmann. Data Integration using Semantic Technology: A use case. In *RuleML*, Seiten 58–66, 2006.

[BB03] A. Borgida und R. Brachman. Conceptual Modeling with Description Logics. In *Description Logic Handbook*, Seiten 349–372, 2003.

[FDES98] D. Fensel, S. Decker, M. Erdmann und R. Studer. Ontobroker in a Nutshell. *Lecture Notes in Computer Science*, 1513:663–664, 1998.

[GGMO03] A. Gangemi, N. Guarino, C. Masolo und A. Oltramari. Sweetening WordNet with DOLCE. *AI Magazine*, 24 (3):13–24., 2003.

[GM03] A. Gangemi und P. Mika. Understanding the Semantic Web through Descriptions and Situations. In *Proc. DOA/CoopIS/ODBASE*, 2003.

[GW02] N. Guarino und C. Welty. Evaluating Ontological Decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, 2002.

[KLW95] M. Kifer, G. Lausen und J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *J. ACM*, 42(4):741–843, 1995.

[MGOS03] C. Masolo, N. Guarino, A. Oltramari und L. Shneider. The WonderWeb Library of Foundational Ontologies. Bericht, 2003.

[Mil04] S. K. Milton. Top-Level Ontology: The Problem with Naturalism. In *Proc. FOIS*, 2004.

[MvH04] D. McGuinness und F. van Harmelen. Web Ontology Language (OWL) Overview. http://www.w3.org/TR/owl-features/, Feb 2004. W3C Recommendation.

[NeO] NeOn Toolkit. http://www.neon-toolkit.org/.

[NIM] NIMS standardization initiative. http://www.fema.gov/emergency/nims/rm/rt.shtm.

[Ont] OntoStudio. http://www.ontoprise.de/.

[SRS$^+$07] P. Smart, A. Russell, N. Shadbolt, M. Schraefel und L. Carr. AKTiveSA: A Technical Demonstrator System For Enhanced Situation Awareness. *Computer Journal*, 50(6):703–716, 2007.

[Vem04] T. Vemmer. *The Management of Mass Casualty Incidends in Germany - From Ramstein to Eschede*. BoD, 2004.