Risk register database to improve organizational resilience and knowledge management

Arto Niemi, Tobias Höbbel, Frank Sill Torres

Department for Resilience of Maritime Systems, Institute for the Protection of Maritime Infrastructures, German Aerospace Center (DLR), Germany. E-mail: arto.niemi@dlr.de

Risk monitoring is a fundamental part of risk management that allows detecting changes that might affect the risk consequences and their likelihood. To be effective, the process requires storing the risk information into a so-called risk register. This paper presents an implementation that is realized in form of a relational database. We present the complete table schema of the database and discuss what motivated the different features. We believe that certain aspects of our schema improve its application in comparison to related works. A noteworthy characteristic is the separation of the risk scenarios and risk analyses in different tables. This feature relates to the fact that the same scenario can be assessed within multiple analyses, in which individual circumstances may result in different risk levels. Moreover, different analyses can apply distinct risk criteria. Thus, the analysis-specific criterion must be stored in the database as a risk matrix. A risk scenario includes an entity or item that is the subject being considered in that scenario. This subject can be a system or organization or a subpart of a system or organization. Due to this reason, the schema allows a hierarchical categorization of entities. The developed schema also employs ideas from the object-oriented programming approach, which allows entities to inherit already defined risk scenarios. This paper further presents a browser based user interface to access the database.

Keywords: Risk register, Relational database, Table schema, Knowledge management, Web application.

1. Introduction

According to Woods (2015), a system's ability for sustained adaptability is one of the core concepts of resilience. This view is backed by the risk management vocabulary standard ISO (2009) that defines resilience solely as the *"adaptive capacity of an organization in a complex and changing environment"*. Over the system's life cycle, changes may occur in any assumptions, boundary conditions, operations conditions, or context of use. When these changes affect the risk level, they trigger the need to adapt. In the risk management context, the detection of changes that might affect the risk consequences and their likelihood is one part of risk monitoring ISO (2009). In this task, the acquired information on changes is employed for reviewing the existing risk controlling and mitigation measures to ensure that they remain sufficient. To be effective, the task requires comparing the former risk assessment results to the present situation. For this purpose, results of risk assessments are recorded in a so-called risk register ISO (2009). However, this is not the only use case for such a register. It can also be employed for expanding, maintaining, and sharing knowledge among different systems or organizations.

This paper presents a risk registry implementation that is realized in form of a relational database (DB). Section 2 presents a short literature review and the motivation for our DB. Section 3 describes the DB schema and what motivated the different features. Section 4 describes the intended use of the DB and shows an User Interface (UI) to access the information, before conclusions in section 5.

2. Background and Motivation

Whipple and Pitblado (2009) show an example from the chemical and petroleum industries that emphasizes the centralized role of the risk registry. The companies process for creating a risk registry is initiated by risk identification, which fills the registry with risk information. The controls to modify the risks are then defined for the highest risks. The value of the register is in communicating the risks to the stakeholders and storing the results to facilitate continuous risk reduction.

The risk registers have also been criticized. Drummond (2011) states that no register can be fully comprehensive or accurate, and by no means listing a risk in a register with a mitigation measure mean that the risk is actually in control. Yet, having all this information listed in a register may create a false sense of security and lead managers unaware of uncertainties. Kutsch and Hall (2010) go further and accuse that sometimes the risk information is treated with deliberate ignorance. In response, Budzier (2011) argues that the issues found by Drummond (2011) are social, cultural, and organizational rather than caused by the risk registers. He sees them as valuable tools to communicate the current understanding of risks within an organization and to prioritize the risk mitiga-

Proceedings of the 31th European Safety and Reliability Conference.
Edited by Bruno Castanier, Marko Cepin, David Bigaud and Christophe Berenguer
 Published by Research Publishing, Singapore
ISBN: 981-973-0000-00-0 :: doi: 10.3850/981-973-0000-00-0_esrel2021-paper˙V4

tion efforts.

Although a risk registry database is a known concept, an article by Patterson and Neailey (2002) found that only a few studies have been made on their development and construction. They referred to an earlier study by Crossland et al. (1998) to state that, at that time, about 78% of the computerized risk registers were developing within individual organizations, and their design was undisclosed. We believe that the design of these registers is rather simple. For example, Whipple and Pitblado (2009) build their register on spreadsheet software. After the article by Patterson and Neailey (2002), only some others have been published on risk registers development and construction. We found an article by Burcar Dunović et al. (2013) and articles linked to an EU-funded TOSCA study, of which Leva et al. (2017) is the latest.

Existing literature gives a rather simple formulation for risk registries. None of the cited examples differ significantly from the description given in Leva et al. (2017), where the core components contain:

Risk ID A unique identification (ID[a]) number for each risk,
Risk Description a concise description or title for the risk,
Risk ranking a quantification of the risk, based on severity and likelihood,
Owner the person responsible for managing the risk and ensuring actions against it are completed,
Actions a list of actions for each risk, and
Dates The date of entry and modification should be held for each risk to assist with reviews. Action target and completion dates should also be included.

The risk ranking is derived based on the likelihood and consequences of the risk. They have predefined categories, as organizations tend to standardize their approaches. Allowing individual analyses to use distinct risk criteria would go against this idea. Also, Patterson and Neailey (2002) consider an individual use case where all risks can be defined using a single criterion for severity and likelihood. The study by Whipple and Pitblado (2009) was in-fact motivated by the need to develop a common risk criterion for an oil company.

One reason why a common risk criteria works in the previous examples was that they considered that each project should have their own risk registry. In slight contrast, Burcar Dunović et al. (2013) presented a system where risk registries are formed for individual projects that are later combined in a central risk registry. Only in their case, it is not clear if the criteria are predefined.

Risk criteria are specific for organizations and use cases. For example, NAVAIR (2005) defines failure rates in relation to flight hours and consequences in terms of personnel injuries, aircraft damage, or mission success. This is different compared to the criteria defined for submarine pipelines, where severity is defined in terms of injuries, and environmental, economic, or political consequences DNV (2013). The acceptable failure probabilities depend partially on the type of fluid carried in the pipeline, as fluid's potential toxicity and flammability will increase the severity of an accident.

In the cited examples, the registry was mainly designed to be used by a single project. There are few key issues that has to be considered in a case where multiple analyses are stored in the registry. The same scenario can be assessed within multiple analyses, in which individual circumstances may result in different risk levels. The analyses may use a specific risk criterion and terminology. There may also be other analysis specific information that is not usually added to a risk analysis. Section 3 presents our implementation of a DB schema that attempts to address these questions, while also employing software engineering methods to aid data storing.

3. Database Schema

This section describes and motivates the different aspects of our table schema that is shown in Fig. 1. Section 3.1 describes the key ideas of the DB. Section 3.2 describes how risk scenarios are stored in the *ScenarioTable*, and what information is stored in *Entity*, *Domain*, and *Example* tables. Section 3.3 introduces the storing of individual analyses and their source documentation. Finally, section 3.4 describes how analysis-specific risk matrices are stored in the DB.

3.1. *Description of key features*

Our schema is designed for storing multiple separate risk assessments. This led us to store the risk scenarios and risk analyses in different tables. As the same scenario can be assessed within multiple analyses, in which individual circumstances may result in different risk levels. The analyses may also use a specific risk criterion. We concluded that trying to present these criteria using some common risk criterion would distort the original information. Especially as organizations have different levels of risk acceptance. So, we decided to store the analyses-specific risk matrices and criteria in the DB. We did not find this concept used in other risk registers. As their main focus was on individual projects and use of distinct criteria in different analyses would not allow forming a standard one for an organization.

[a]This acronym is later is used for marking the primary and foreign key values in our schema. In a relational database, a primary key uniquely identifies each record in a table.

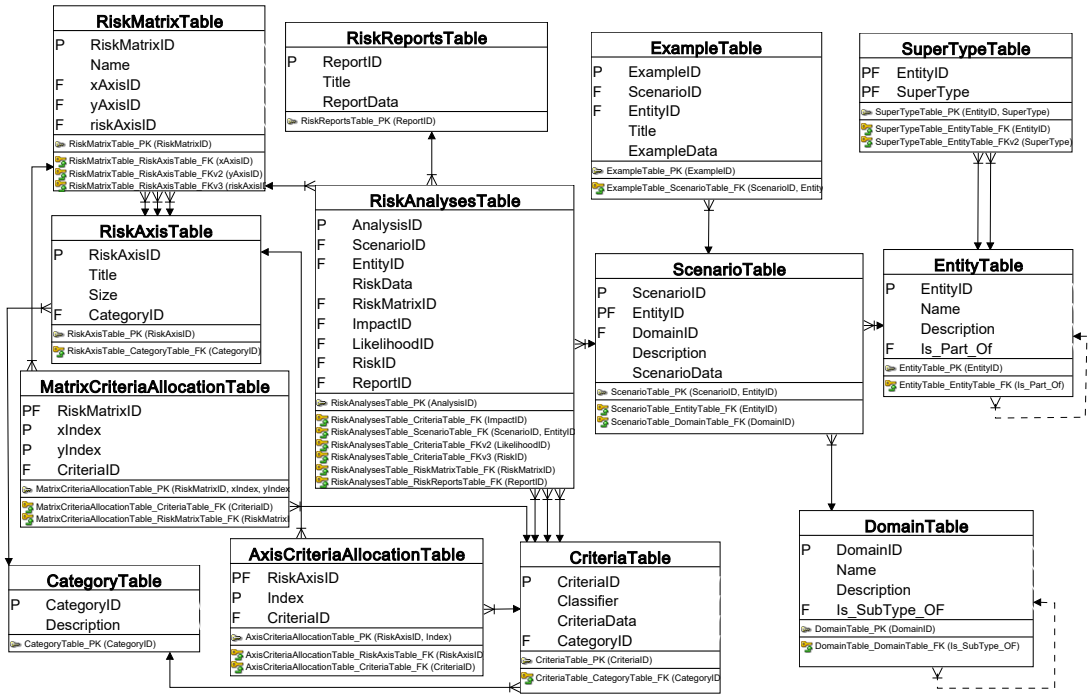

Fig. 1. Table schema of the proposed risk register database. The letter P stands for the primary key and F for a foreign key. The foreign key relations are also defined with arrows.

Our schema is mainly for knowledge management rather than project management, information like the risk owner, mitigation plan, or project-specific dates are not included as specified columns. Different analyses are too distinct for defining tables that contain all the required columns for defining the information. However, this information can still be added to the DB, as many tables in the schema allow users to store the information in the JavaScript Object Notation (JSON) data format. It allows among other things storing information in freely defined name/value pairs Bray (2017). We also provide a link to the source documentation of an analysis, which may contain additional information.

The schema further employs some technical methods to aid data storing. For example, a risk scenario includes an entity or item that is the subject being considered in a risk scenario. This subject can be a system or organization or a subpart of a system or organization. Due to this reason, the schema allows a hierarchical categorization of entities. The developed schema also employs ideas from the object-oriented programming approach Hillar (2015), which allows entities to inherit already defined risk scenarios. This can also be used in the case where various terms are used to describe similar entity in different organizations. Section 3.2.2 shows an example where *"boat"* and *"ship"* are both defined as *"watercraft"*. Searching for one term allows user to find the related ones. This approach can be further used for defining synonyms by having the synonym terms to inherit the risk scenarios defined for the main term.

3.2. *Risk scenarios*

3.2.1. *Schema*

In our schema, the principal components of a risk scenario are the entity or item that is the subject being considered in a risk scenario, and the description of the scenario. A simple example is *"Ship sinks."*, where the ship is the subject and sinking is the risk. This information is stored in the *ScenarioTable*, which contains the following columns:

ScenarioID forms the primary key with the **EntityID**,
EntityID foreign key from the *EntityTable*,
DomainID foreign key from the *DomainTable*,
Description free text that describes the risk scenario, and
ScenarioData data related to the scenario that is stored in the JSON format.

The **ScenarioID** and the **EntityID** form a composite primary key. This feature is related to

the object-oriented programming approach Hillar (2015). In this approach, an object may have a super-type from which the object inherits features that are defined for the super-type. For example, ships and boats are both watercraft and will inherit the sinking scenario if it is defined for the watercraft. In some cases, the object must alter the definition that has been made for the super-type. In these cases, the object-specific definition overrides the super-type's definition. In the schema, if an entity has a super-type, it will automatically inherit the risk scenarios that have been defined for the super-type. If a risk scenario is significantly different from the super-type's scenario, the inherited scenario can be overridden. This is done by creating a primary key with a **ScenarioID** that is defined for the super-type in combination with the entity's **EntityID**. This way, the composite primary key enables entities to override super-type's risk scenarios. The inheritance aspects are clarified in section 3.2.2.

The *EntityTable* describes individual entities and allows defining their hierarchy with a technique called adjacency list Celko (2012). The table contains the following columns:

EntityID the primary key,
Name name of the entity,
Description free text that describes the entity, and
Is_Part_Of a *"foreign"* key from *EntityTable* that allows defining hierarchies.

The **Is_Part_Of** column can contain the **EntityID** of another entity of which the defined entity is a part of. For example, a rudder is a part of a ship.

The super-types are defined in the separate *SuperTypeTable*. It contains two columns: **EntityID**, and **SuperType**, which both are foreign keys from the *EntityTable*. The columns also form a composite primary key to ensure that each relation is unique. Using a separate table enables multiple inheritance of risk scenarios from several super-types.

The *DomainTable* defines what type of risk a scenario considers. For instance, a domain can be *personnel safety* or *security*. The table contains columns:

DomainID the primary key,
Name name of the domain,
Description free text that describes the domain, and
Is_SubType_Of a *"foreign"* key from the *DomainTable* that allows defining hierarchies.

Similar to the *EntityTable*, the **Is_SubType_Of** column allows defining hierarchies with the adjacency list technique. For example, *cyber-security* is a sub-type of the *security* domain.

We considered that a short technical definition of risk scenarios may be hard to comprehend. Especially, if a person is not familiar with different risk scenarios in the DB. Therefore, we borrowed an idea from the pedagogy that values the use of examples to aid learning Oliveira and Brown (2016), and added a specific table for examples where a risk scenario has been realized. These examples can be news items or accident investigation reports that help to connect risk scenarios to real life.

Examples of risk scenarios are stored in the *ExampleTable*. Having this information in a separate table allows defining multiple examples for a single risk scenario. The table has columns:

ExampleID the primary key,
ScenarioID a foreign key from the *ScenarioTable*,
EntityID also a foreign key from the *ScenarioTable*, as that table has a composite primary key,
Title title of the report, and
ExampleData bibliographical information in JSON format.

We cannot foresee what bibliographical information is available for different reports. Therefore, we allow users to input this information in the JSON form. One can imagine that the information could contain similar fields as the BibTeX formatting system Patashnik (1988).

3.2.2. *Example of risk scenario inheritance*

This example clarifies the concept of entities inheriting risk scenarios from their super-types. Watercraft, ships, and boats can be organized in a class hierarchy shown in Fig. 2. This hierarchy can be defined in the *SuperTypeTable* as shown in Table 1. A simplified *RiskScenarioTable* shown in Table 2 demonstrates how an entity can override risk scenario description. By default, all the subclasses of the watercraft inherit the sinking scenario. The cargo and passenger ships override the scenario description with a more detailed one. The description for the RMS[b] Titanic further overrides the passenger ship's scenario description.

Although the Titanic sank relatively slowly, many of the passengers and crew had little chance to survive. This aspect is significantly different from most modern accidents. As two years after the disaster, the First International Conference on Safety of Life at Sea (SOLAS) was held and introduced several safety regulations, such as defining the sufficient number of lifeboats for all persons on board Phillips and Sirkar (2012). Had these regulations been in place before the disaster, the outcome could have been much less severe.

[b]Royal Mail Ship

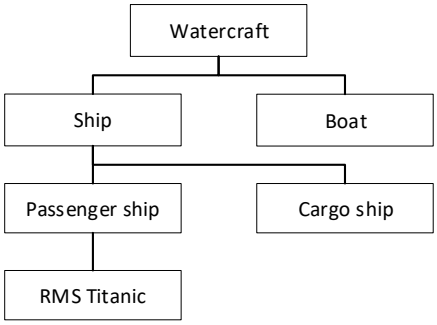

Fig. 2. Example of a class hierarchy for watercraft.

Table 1. The example class hierarchy in the *SuperTypeTable*. Unlike in the actual schema, the entity names are shown instead of the ID numbers.

Entity	SuperType
Watercraft	
Boat	Watercraft
Ship	Watercraft
Cargo ship	Ship
Passenger ship	Ship
RMS Titanic	Passenger ship

3.3. *Individual risk analyses*

In our schema, a risk analysis extends a risk scenario by determining how likely it is and what are its consequences. This information is then used for assigning the risk level. A risk scenario can be considered in multiple analyses, which may result in different findings. Therefore, these results must be stored in a different table from the scenarios. The *RiskAnalysesTable* contains the following columns:

AnalysisID the primary key,
ScenarioID a foreign key from the *ScenarioTable*,
EntityID also a foreign key from the *ScenarioTable*, as that table has a composite primary key,
RiskData information in JSON format,
RiskMatrixID a foreign key from the *RiskMatrixTable*,
ImpactID a foreign key from the *CriteriaTable*,
LikelihoodID a foreign key from the *CriteriaTable*,
RiskID a foreign key from the *CriteriaTable*, and
ReportID a foreign key from the *RiskReportsTable*.

The **ScenarioID** and **EntityID** link the analysis to the scenario that was described in section 3.2. The **RiskMatrixID** and foreign keys from the *CriteriaTable* link the analysis to a risk matrix, which will be introduced in section 3.4. Although, a risk matrix is linked to a defined criteria, this criteria may be used in several other matrices. Therefore, the **RiskMatrixID** is essential for determining the specific risk matrix that is used in an analysis.

The **ReportID** links analysis to the *RiskReportsTable*. In our view, the DB does not have to provide the full information of individual analyses. This can be provided in the source documents. In any case, it is important to know where the analysis information originated from. Thus, storing the details of sources provides traceability.

The *RiskReportsTable* is quite similar to the *ExampleTable* that was introduced in section 3.2. It contains columns:

AnalysisID the primary key,
Title title of the report, and
ReportData bibliographical or other information in JSON format.

The table does not contain the **AnalysisID**, as we foresee that there is just one primary report for each analysis that is defined in the *RiskAnalysesTable*.

3.4. *Risk matrices*

This section describes how risk matrices can be stored in the DB. We use a simple matrix shown in Fig. 3 is used for describing what data is stored in each table.

A risk matrix is in the *RiskMatrixTable*, which contains columns:

RiskMatrixID the primary key,
Name name of the matrix,
xAxisID foreign key from the *RiskAxisTable* for the x-axis,
yAxisID foreign key from the *RiskAxisTable* for the y-axis, and
riskAxisID foreign key from the *RiskAxisTable* for the risk categories.

Considering the example matrix, its name could be the *"Simple matrix"*, and in this case, the x-axis is for consequences and the y-axis for likelihoods.

Axes are defined in the *RiskAxisTable*. It has the following columns:

RiskAxisID the primary key,
Title Title of the axis,
Size the number of classification criteria of the axis, and
CategoryID foreign key from the *CategoryTable*.

Table 2. Example of a simplyfied *RiskScenarioTable*, where Entities override the inherited risk scenarios with more detailed descriptions. Again for clarity, the names are shown instead of the ID numbers.

Scenario	Entity	Description
Sinking	Watercraft	Watercraft sinks
Sinking	Cargo ship	Cargo ship sinks endagering crew and cargo
Sinking	Passenger ship	Passenger ship sinks endangering the crew and passengers
Sinking	RMS Titanic	Passenger ship sinking before SOLAS 1914 regulations

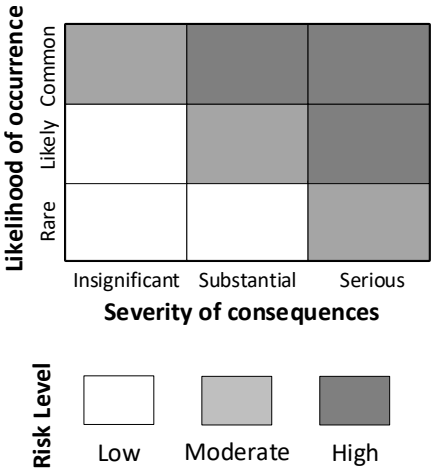

Fig. 3. Example of a risk matrix.

The number of classifiers for each axis can be freely defined, but in this example, all of the axes have three classifiers. So, their size is three. The axis titles are displayed in Fig. 3. For example, the title of the risk axis is *"Risk Level"*.

The category determines the types of an axis. This is done in the *CategoryTable* that has two columns: **CategoryID** and **Description**. The category can be *"likelihood"*, *"consequences"*, or *"risk"*.

Defining the category is important as the classifier criteria are defined independently from axes in the *CriteriaTable*. This table contains columns:

CriteriaID the primary key,
Classifier text that describes the criterion,
CriteriaData data in JSON form, and
CategoryID foreign key from the *CategoryTable*.

The classifiers are shown when a matrix is drawn. In Fig. 3, *"Rare"*, *"Likely"*, and *"Common"* are the classifiers for the y-axis's criteria. The schema does not have a specific column for the color of the risk levels or specific descriptions. These will be stored in the JSON data.

The criteria are assigned to axes in the *AxisCriteriaAllocationTable*, which contains columns:

RiskAxisID a foreign key from the *RiskAxisTable*,
index the axis location of the allocated criterion, and
CriteriaID a foreign key from the *CriteriaTable*.

The primary key is defined as a combination of **RiskAxisID** and **index** to ensure that each index location can have only one allocated criterion. The criteria must be allocated to both coordinate axes and the risk axis. The former allows drawing the risk axis as seen on the bottom of Fig. 3.

The criteria have to be further assigned to each element of the matrix to be able to display it. This allocation is done in the *MatrixCriteriaAllocationTable*. It contains columns:

RisksMatrixID a foreign key from the *RiskAxisTable*,
xIndex the axis location of the allocated criterion,
yIndex the axis location of the allocated criterion, and
CriteriaID a foreign key from the *CriteriaTable*.

The primary key is a composite of **RisksMatrixID** and the **indexes**. The purpose is to ensure that each element of the risk matrix can have only one assigned criterion.

The presented tables allow storing risk matrices in the DB. The high degrees of freedom to customize the size of the matrix and definitions require a quite complex schema to define the features.

4. Intended Use of the Database

4.1. *Searching information*

This section describes how information stored in the DB can be browsed to attain knowledge of identified risks. Authors foresee three main use cases. In the first case, a user searches for information on specific entities, such as ships or offshore wind turbines. This requires an ability to query the *EntityTable* to see if the searched term

is mentioned the **Name** or the **Description** of an entity.

Once the user selects an entity, the UI shall show all the scenarios it is involved in. This requires showing 1) the scenarios that the entity inherits from its super-types, and 2) the scenarios defined for the sub-items of an entity, which are defined as parts of the entity in the *EntityTable*. It is also important to note that a scenario may occur where the entity is not the subject of the scenario, but is related to it. Showing this information for a user requires querying the **Descriptions** in the *ScenarioTable* in case of the **Name** of the entity is mentioned in the text.

The second and third ways to access the information involve the stored documentation. A user may be interested to see what risk reports are stored in the DB and what analyses these reports contain. Browsing the reports requires querying the *RiskReportsTable* and showing the analyses requires querying the *RiskAnalysesTable* with a specified **ReportID**. A user may also browse through the stored examples and wants to see what information the DB contains for the depicted scenario. Browsing the examples requires querying the *ExampleTable*, which contains the scenario specific **ScenarioID** and **EntityId** to access more information.

4.2. *Browser based user interface*

This section shows excerpts from the UI used for accessing the information stored in the DB. This also demonstrates the ability to store information from multiple sources. Our examples are from a master's thesis by Mielniczek (2019) that studies occupational safety of offshore wind farms, a risk assessment of sub-sea power cables by Carbon Trust (2015), and the National Risk Register by HM Cabinet Office (2020).

Fig. 4 shows how the data is stored in the *RiskAnalysesTable*. For the case from Cabinet Office (2020), the UI shows an extended information including the risk matrix. At the time of writing this paper, the UI is not fully finished, and the authors are experimenting on what information should be shown for the user in different views.

The issue caused by different analyses using distinct risk criteria is clearly shown by Fig. 4. Mielniczek (2019) uses both numbers and text to define the criteria. Here we opted to show the text definitions. Carbon Trust (2015) presents only numeric definitions that are not otherwise defined in the source document. Therefore, the numbers are shown. Finally, the national risk register by Cabinet Office (2020) uses probabilities to define the likelihoods. The risk levels are not defined, but consequences have eight different definitions using different scales of impact.

In the case of the *"Level D"*, the consequence can be defined for examples on the economic scale as one to ten billion pounds impact, or in terms of electricity supply as a major disruption for one million people that last more than 18 hours. The document further uses several pages to describe each risk scenario. These kinds of long descriptions are not intended to be stored in our DB. For this reason the DB stores the information on the source document, which may contain this kind of information.

5. Conclusions

This paper presented a database schema for a risk register and a browser based UI to access the risk information. The examples of risk registers found in the literature survey were mainly designed for risk management in projects. In contrast, our DB is mainly intended for knowledge management and storing multiple risk analyses. One consequence of this is that the DB has to store several distinct risk criteria and multiple analyses for a same scenario. These requirements affected the schema design. Our work will continue by improving the UI and by employing the DB to its intended use in our institution.

Acknowledgements

Authors are thankful for Dr. Bartosz Skobiej for his helpful comments during this research.

References

Bray, T. (2017). The JavaScript Object Notation (JSON) data interchange format. RFC 8259, Internet Engineering Task Force.

Budzier, A. (2011). The risk of risk registers – managing risk is managing discourse not tools. *J. Inf. Technol. 26*(4), 274–276.

Burcar Dunović, I., M. Radujković, and M. Vukomanović (2013). Risk register developement and implementation for construction projects. *J. Croat. Assoc. Civ. Eng. 65*(1), 23–35.

Cabinet Office (2020). *National Risk Register* (2020 ed.). HM Government.

Carbon Trust (2015). *Cable burial risk assessment methodology, Guidance for the preparation of cable burial depth of lowering specification*. Number CTC835. The Carbon Trust.

Celko, J. (2012). *Joe Celko's trees and hierarchies in SQL for smarties* (Second ed.). Morgan Kaufmann.

Crossland, R., C. A. McMhahon, and J. H. Sims Williams (1998). *Survey of current practices in managing design risk*. University of Bristol.

DNV (2013). Submarine pipeline systems. Offshore standard DNV-OS-F101, Det Norske Veritas AS.

Drummond, H. (2011). MIS and illusions of control: An analysis of the risks of risk management. *J. Inf. Technol. 26*(4), 259–267.

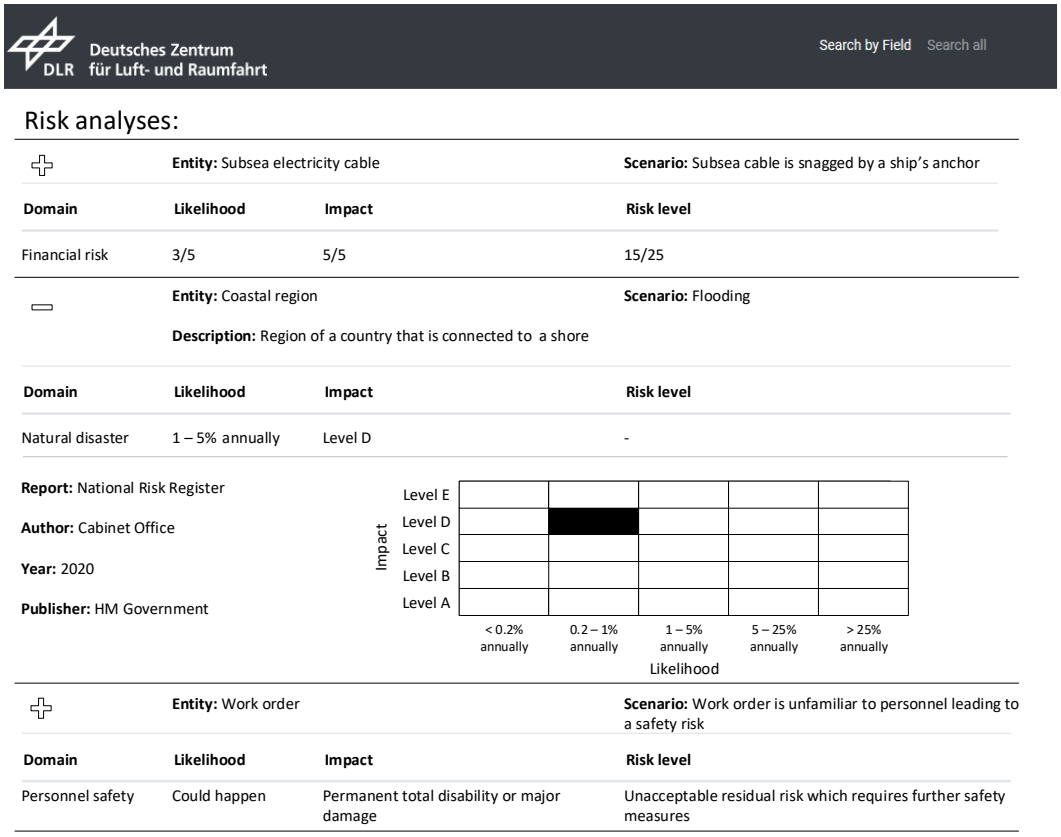

Fig. 4. Excerpt from browser based user interface to access the data in our risk registry database. Figure shows data from the *RiskAnalysesTable*, with examples from the reports Carbon Trust (2015), Cabinet Office (2020), and Mielniczek (2019). An extended information is shown for the case from Cabinet Office (2020). At the time of writing this paper the UI is not fully finished, and what information should shown in different views is being debated.

Hillar, G. C. (2015). *Learning Object-Oriented Programming*. Packt Publishing, Limited.

ISO (2009). Risk management - Vocabulary. ISO Guide 73:2009, International Organization for Standardization.

Kutsch, E. and M. Hall (2010). Deliberate ignorance in project risk management. *Int. J. Proj. Manag. 28*(3), 245–255.

Leva, M. C., N. Balfe, B. McAleer, and M. Rocke (2017). Risk registers: Structuring data collection to develop risk intelligence. *Saf. Sci. 100*, 143–156.

Mielniczek, J. (2019). Quantification of occupational safety for offshore wind farms. Master's thesis, Jade University of Applied Sciences.

NAVAIR (2005). Guidelines for the naval aviation reliability-centered maintenance process. Management manual NAVAIR 00-25-403, Naval Air Systems Command.

Oliveira, A. W. and A. O. Brown (2016). Exemplification in science instruction: Teaching and learning through examples. *J. Res. Sci. Teach. 53*(5), 737–767.

Patashnik, O. (1988). *BibTeXing*.

Patterson, F. D. and K. Neailey (2002). A risk register database system to aid the management of project risk. *Int. J. Proj. Manag. 20*(5), 365–374.

Phillips, C. and J. Sirkar (2012). The International Conference on Safety of Life at Sea, 1914: The history and the ongoing mission. *Coast Guard J. Saf. Secur. Sea: Proc. Mar. Saf. Secur. Counc. 69*(2), 27–28.

Whipple, T. and R. Pitblado (2009). Applied risk-based process safety: A consolidated risk register and focus on risk communication. *Process Saf. Prog. 29*(1), 39–46.

Woods, D. D. (2015). Four concepts for resilience and the implications for the future of resilience engineering. *Reliab. Eng. Syst. Saf. 141*, 5–9.