Chapter 6
Using Ontologies and Machine Learning

O. Daramola, T. Stålhane, I. Omoronyia, and G. Sindre

Abstract Safety analysis (SA) procedures, such as hazard and operability analysis (HazOp) and failure mode and effect analysis (FMEA), are generally regarded as repetitious, time consuming, costly and require a lot of human involvement. Previous efforts have targeted automated support for SA at the design stage of system development. However, studies have shown that the cost of correcting a safety error is much higher when done at the later stages than the early stages of system development. Hence, relative to previous approaches, this chapter presents an approach for hazard identification (HazId) based on requirements and reuse-oriented safety analysis. The approach offers a convenient starting point for the identification of potential system safety concerns from the RE phase of development. It ensures that knowledge contained in both the requirements document and previously documented HazOp projects can be leveraged in order to attain a reduction in the cost of SA by using established technologies such as ontology, case-based reasoning (CBR), and natural language processing (NLP). The approach is supported by a prototype tool, which was assessed by conducting a preliminary evaluation. The results indicate that the approach enables reuse of experience in conducting safety analysis, provides a sound basis for early identification of system hazards when used with a good domain ontology and is potentially suitable for application in practice by experts.

O. Daramola (✉)
Covenant University, Ota, Nigeria
e-mail: olawande.daramola@covenantuniversity.edu.ng

T. Stålhane • G. Sindre
NTNU, Trondheim, Norway
e-mail: stalhane@idi.ntnu.no; guttors@idi.ntnu.no

I. Omoronyia
University of Glasgow, Lanarkshire, Scotland
e-mail: Inah.Omoronyia@glasgow.ac.uk

6.1 Introduction

Safety analysis (SA) embraces all of the hazard identification (HazId), risk and safety assessment activities involved in the development of safety-critical embedded systems. The goal of SA is to influence safety-critical system design by conducting several types of safety procedures in order to identify potential system hazards and risks and to mitigate them to acceptable levels before a system is certified. Safety analysis procedures, such as hazard and operability analysis (HazOp) and failure mode and effect analysis (FMEA), are generally regarded as repetitious, time consuming, costly and require a lot of human involvement [1–3]. Although human expertise is irreplaceable in the conduct of effective SA procedures at the moment, there is a need to reduce the amount of human effort and cost of SA. Previous efforts to address this problem have been based largely on expert system approaches, which target automated support for SA from the design stage of system development [1, 4]. However, studies [5, 6] have shown that the cost of correcting a safety error is much higher when done at the later stages than the early stages of system development. Since requirements engineering (RE) precedes system design, it provides a convenient starting point for the identification of potential safety concerns of a system if the knowledge contained in requirement documents can be extracted and used as the initial basis for SA. Hence, tool support for SA at the RE phase will be more beneficial for attaining a reduction in the cost of hazard identification and hazard mitigation.

HazOp is one of the prominent safety analysis techniques [4]. HazOp is used to study hazards and operability problems by investigating the effects of deviations from prescribed design intent in order to mitigate the occurrence of adverse consequences. It involves early discovery of potential system hazards and operation problems and recommendation of appropriate safeguard mechanisms by a team of experts.

However, HazOp is a time consuming, costly and a largely human-centred process [1, 3, 6]. The HazOp process is essentially subjective, relying on the professional experience, expertise and creativity of the team members involved. Some of the crucial challenges of HazOp which are still open research issues are: (1) how to reduce the level of subjectivity, (2) how to reduce the amount of human effort, (3) how to promote reuse of valuable knowledge gained in previous HazOp studies and (4) how to facilitate transfer of HazOp experiences among HazOp teams [3, 7]. These challenges motivate the need for a framework that could enable early identification of hazards and reuse-oriented HazOp analysis. The first objective of this work is to provide a decision support tool that could assist the human expert in the process of identifying potential safety concerns that are contained in the requirements document. The second is to create a platform for the reuse of knowledge from previous HazOp studies in subsequent projects, in order to reduce the amount of human effort needed while conducting HazOp. This work would be useful in the safety analysis of product line systems or variant systems, where the systems share a significant degree of commonality. Also, the approach could be valuable in the context of system development models that are iterative or incremental in nature where there is a need to continually revise requirements and design specifications during the period of development. Our focus on HazOp stems from the interests of the CESAR project[1] that we are currently involved in.

We have adopted an approach that combines three technologies to realise the stated objectives of this work, namely:

- Case-based reasoning (CBR), which is a pattern-based problem solving paradigm that enables the reuse of previously gained knowledge in resolving a new case [8]
- Ontology, which is the semantic representation of the shared formal conceptualization of a domain that provides a platform for the standardisation of terms and vocabulary in the domain [9]
- Natural language processing (NLP) which is the processing and analysis of natural language text [10]

A prototype tool called KROSA (knowledge reuse-oriented safety analysis) that demonstrates the novel integration of these three technologies has been created to validate our approach. The unique contribution of this work is the integration of ontology and machine learning technologies into a framework that enables the identification of hazards from requirements and reduction of effort needed for HazOp through knowledge reuse. In this chapter, we present a description of the proposed framework and the evaluation of the prototype tool by an experiment and opinions provided by domain experts at ABB Norway.

The rest of this chapter is organised as follows: Sect. 6.2 presents the background for the context of this chapter, while Sect. 6.3 describes a HazOp problem example and how tool support can be provided for HazId based on requirements. In Sect. 6.4, we give a description of the KROSA framework and how it can be used for HazOp. Section 6.5 presents the evaluation procedure used for assessing the KROSA tool, while Sect. 6.6 discusses the results of the evaluation and the threats to validity of results. In Sect. 6.7, we review some closely related work, and the chapter is concluded in Sect. 6.8 with a brief note and indication of our future research plans.

6.2 Background

In this section, we give a brief overview of the general HazOp process and the key technologies that are relevant to this work.

6.2.1 Overview of the HazOp Process

A hazard and operability study (HazOp) is a structured and semiformalised team-based procedure that focuses on the study of a system under design, in order to identify and evaluate potential hazards that may constitute a risk to personnel or

[1] http://www.cesarproject.eu

equipment or prevent efficient operation of the system. A HazOp study is undertaken by a HazOp team through a series of brainstorming sessions in order to stimulate creativity used to reveal potential hazards in the system and their cause–effect relationships [1, 4]. HazOp is based on the assumption that a problem can only arise when a system deviates from its design and operational intents.

Hence, the HazOp study entails a detailed walkthrough of the process and instrumentation diagram models of a system to spot every likely deviation from its intended operation using a set of *guidewords*. Generally, guidewords represent variations of known system parameters that may cause deviation from design intentions. They are chosen and interpreted based on particular design representation and context. Examples include no, not, more, less, before, after, late, too often and early. Examples of parameter–guidewords pairs include arrive late, arrive early, no flow, not sent and sent after. Guidewords are carefully selected to stimulate reasoning about all potential system hazards. A point of observation pertaining to a system or process that can be a source of a potential hazard is called a *study node*. As each deviation is derived, the HazOp team discusses potential causes, consequences and safeguards and recommend appropriate control actions to forestall or mitigate its occurrence.

Typically, it takes about 1–8 weeks for a HazOp team with 4–8 members to conduct a HazOp, depending on the size and complexity of the system in question. It is widely accepted that HazOp analysis is an extremely time-consuming process [1, 3, 4]. More on the procedure of HazOp study and ideals of HazOp team, membership composition can be found in [11].

6.2.2 Case-Based Reasoning (CBR)

CBR is an instance-based machine learning paradigm that emulates the human reasoning process of solving problems based on past experiences. In CBR, problems are modelled as abstraction called *cases* which consist of the problem part and the solution part. The CBR life cycle [8] is a four-stage process that consists of (1) *case retrieval* – where old cases that are similar to a new case are identified by comparing the problem parts of the old cases and that of the new case using a similarity metric; (2) *case reuse* – which entails applying the solution part of the most relevant old case or group of old cases to the new case, and this may also involve adaptation of the old solutions to fit the new case; (3) *case revision* – where the reused solution is tested for appropriateness in the new case, and if need be, the reused solution is revised to fit the new case; and (4) *case retention* – which entails storing a solved case in the case base (repository) for future reuse. CBR provides a mechanism of organising, storing and reusing an organisation's memory or experiences. As such, it offers a credible model of experience-based problem solving once relevant cases exist [12]. The CBR paradigm is considered particularly relevant to the context of HazOp because of its potential to support the acquisition, retrieval, reuse and retention of knowledge, which provides a basis for documented experiences from previous HazOp studies to be leveraged in subsequent HazOp projects.

6.2.3 Ontology

Ontology which is a shared formal conceptualization of a domain is a key technology to shaping and exploiting information for the effective management of knowledge that pertains to specific domains [13]. Ontologies have human and machine-readable semantics that allow definition of semantic relationships between entities and inference of knowledge through reasoning at runtime. According to [14], ontologies have the capability to (1) enable knowledge reuse, (2) ensure better understanding of a knowledge area, (3) support analysis of the structure of knowledge, (4) foster understanding of available knowledge in a domain and (5) provide embedded knowledge for an application that can be used by machines. Ontology is considered relevant to the HazOp problem because of its potential to facilitate (1) formalised semantic description of relevant domain knowledge for identification of system hazards, (2) interoperable transmission of knowledge among HazOp teams and (3) knowledge reuse while conducting HazOp.

6.2.4 Natural Language Processing (NLP)

NLP is concerned with the process of extracting meaningful information from natural language text through the use of statistical machine learning algorithms [10]. In NLP, machine learning algorithms automatically learn rules through the analysis of large *corpora* of real-world examples. A *corpus* (plural, “corpora”) is a set of documents that have been manually annotated with the correct values to be learned. The learned rules are then used to classify words into various word categories (part of speech) following the supervised learning model. Key NLP operations include sentence tokenisation, part-of-speech tagging, coreference resolution, anaphora resolution, named-entity recognition and morphology analysis. NLP is a necessity for automated requirements analysis because requirements are mostly written as natural language text. Therefore, our approach uses NLP in combination with ontology to enable the extraction of useful knowledge from natural language requirement documents for the early identification of potential system hazards.

6.2.5 Knowledge Management in Requirements Engineering

In recent times, the application of knowledge management technologies such as ontologies, NLP and CBR has gained momentum in requirements engineering. In [15], the SoftWiki approach was reported as a way of semantifying requirements engineering. According to the authors, semantification of RE entails representing each requirement as a unique instance of the Semantic Web having its own URI such that spatially distributed stakeholders – including developers and users – can collect, semantically enrich, classify and aggregate requirements within the context of collaborative software development. The approach uses the SoftWiki Ontology for Requirements Engineering (SWORE) to facilitate the semantification process. Similarly, [14] gave an elaborate overview of how ontologies can be applied in collaborative software development and the vision of a software engineering Semantic Web.

In [16], a framework for requirements elicitation using ontology reasoning was proposed. NLP was used to parse initial requirements to obtain key concepts that can be mapped to functions in the domain ontology. Thereafter, the rules and relations among functions in the ontology were used to reason about errors and potential requirements. Other research efforts where ontologies have been applied for requirements elicitation and analysis include [17] where a domain ontology and requirements meta-model were used to elicit and define textual requirements; in [18], an approach for goal-oriented and ontology-driven requirements elicitation (GOORE) was proposed. In GOORE, the knowledge of a specific domain is represented as an ontology, which is then used for goal-oriented requirements analysis.

In [19], a perspective for the application of CBR for requirements engineering was provided. Also, [12] gave a detailed account of the probable applications of CBR in software engineering in the aspects of prediction and reuse. In [20], CBR was used to evaluate the requirements quality by referring to previously stored software requirements quality analysis cases (past experiences) in order to ensure that the quality of the prepared SRS is acceptable, while [21] proposed a framework for managing implicit requirements by using a combination of ontology and CBR. All of these efforts indicate an increasing interest in the application of ontology, NLP and CBR as knowledge management technologies in requirements engineering.

6.3 Simplified Steam Boiler Example

The steam boiler system is a simplified version of an industrial steam boiler, developed as a first pilot system for testing CESAR concepts. In order to have a simple system, important components such as the feeding tank and the blow down valve are left out.

The functional requirements of the steam boiler are as follows:

1. The steam boiler shall deliver steam at a predefined, constant pressure to an industrial process.
2. Steam is produced by heating water using an electric heating element.
3. The steam pressure is controlled by regulating the temperature setting on the heating element thermostat.
4. The water level in the tank is controlled by a feeding pump which pumps water into the tank via a non-return valve.
5. The safety of the steam boiler is taken care of by a safety valve that opens to air. The release pressure for the safety valve is fixed, based on the boiler’s strength.
6. The system shall be safety integrity level two (SIL2) certifiable.

In the CESAR project, we have embraced the notion of requirements boilerplates[2] which stems from the work in [22, 23] for writing requirements in semiformalised form. A boilerplate is a textual template for requirements specification that is based on predefined patterns, which reduces the level of inconsistency in the way requirements are expressed. We have also introduced additional requirement boilerplates patterns that are considered well suited for embedded systems requirements.

For the steam boiler example, we will now use the following predefined sample boilerplates[2]:

BP1: **The** < system > **shall** < action>
BP2: **The** < system > **shall be able to** < action > **using** < system>
BP3: **If** < condition>, **the** < system > **shall** < action>

The functional requirements of the steam boiler can then be transformed to a semiformal form as follows:

R1: The *< steam boiler >* ***shall be able to*** *< deliver > [<steam >* ***to*** *< an industrial process>]* – BP1
R2: The *< steam boiler >* ***shall be able to*** *< produce > [<steam >* ***using*** *(<electrical > <heating element>)]* – BP2
R3: The *< steam boiler >* ***shall be able to*** *< control > [<steam pressure >* ***using*** *(<thermostat > of < electrical > <heating element>)]* – BP2
R4: The *< steam boiler >* ***shall be able to*** *< control > [<water level >* ***using*** *(<feeding pump>)]* – BP2
R5: The *< feeding pump >* ***shall be able to*** *< deliver > [<water >* ***using*** *(<non-return valve>)]* – BP2
R6: If *[<steam pressure >* ***greater than*** *< critical pressure level>]****, the*** *< steam boiler >* ***shall*** *[<open > <safety valve>]* – BP3

6.3.1 Preliminary HazOp (PHA) for Steam Boiler

Usually, based on a concept diagram for system – say a steam boiler – a team of experts would run a PHA by brainstorming on specific requirements and components of the system in order to identify potential hazards that may arise from possible deviations from the design intent of the steam boiler. The result of such a PHA for a steam boiler system would be a manually generated preliminary HazOp table. A small part of such a table is shown in Table 6.1.

[2] www.requirementsengineering.info/boilerplates.htm

Table 6.1 Preliminary HazOp table for a steam boiler

Req.	Element (study node)	Guideword	Hazard	Cause	Main effect	Preventive action
R2	Water tank (heating element)	Temperature too hot	The water tank is too hot	Too little water and too much heat (sensor, control, actuator, connections)	Tank gets hot/fire	Turn off the heat
						Add water?
R3	Water tank	Pressure too high	Too high pressure in the water tank (R3)	Not able to turn off the heating (sensor, control, actuator, connections)	Boiler explodes	Safety valve
				Feeding pump failure (too strong)	Boiler rupture	Turn off the heat
						Turn off power to the feed pump
R4	Feeding pump	Water level too high	Too high water level (R4)	Water-level regulation failure (sensor, control, actuator, connections)	Water to the process	Pump emergency stop
R5	Feeding pump	Non-return valve stuck	Too high pressure in the feed pipe (R5)	Non-return valve failure	Release boiling water to the water supply	Two non-return valves in series
						Emergency valve for releasing pressure

6.3.2 *Tool Support for HazId Based on Requirements*

Our objective in this work is to provide tool-based support for HazId based on requirements – which is usually a costly manual procedure – such that:

1. Requirement documents can be analysed semantically using a combination of shallow NLP and domain knowledge as contained in the domain ontology, to identify potential system hazards automatically. Hence using the steam boiler ontology (see Fig. 6.2), columns 1 and 2 of Table 6.1 – a HazOp table for the steam boiler system – can be automatically generated.
2. The user is able to partially or totally reuse relevant parts of previously documented HazOp projects in order to generate causes, effect, safeguards and appropriate control actions for each system hazard that has been identified – generate data for columns 2–5 of specific hazards (study node) in Table 6.1.

With this proposed approach, we aim to provide relevant tool support for the HazOp experts so as to reduce the amount of effort needed and also to offer a good starting point for HazOp in instances where there is paucity of experts. We will now describe the architecture of our approach in the next section.

6.4 The KROSA Framework

The architectural framework of our proposed approach is an integration of the three core technologies NLP, CBR and ontology. A view of the architecture is presented in Fig. 6.1. The core system functionalities are depicted as rectangular boxes, while the logic, data and knowledge artefacts that enable core system functionalities are depicted using oval boxes. A detailed description of the KROSA framework is given in the following.

6.4.1 *Knowledge Representation and Extraction*

In this section, we describe the parts of the KROSA architecture that deals with knowledge representation and extraction.

(a) Data Preprocessing

The input to the framework is a preprocessed requirements document. Preprocessing is a manual procedure that ensures that source documents are transformed into a form that is suitable for the framework. It entails extraction of requirements in form of sentences from source documents, extracting sentences that define system requirements and replacing information conveyed in figures, diagrams and tables with equivalent sentences. Also, the requirements could be expressed in

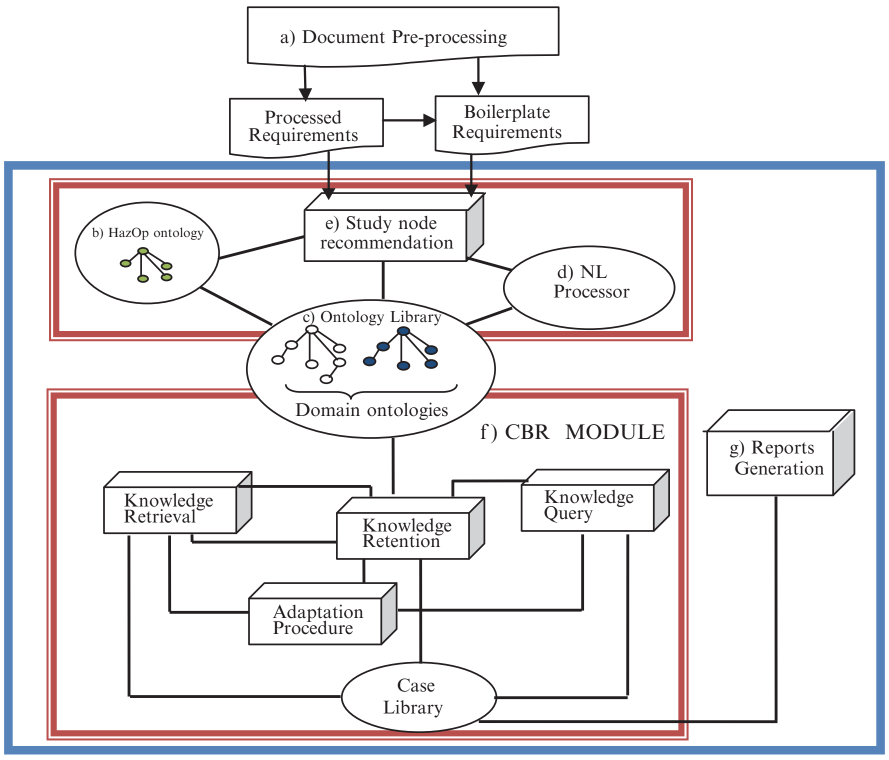


Fig. 6.1 Architectural framework for reuse-oriented HAZOP

semiformalised way using requirement boilerplates. Boilerplate requirements will also be more susceptible to treatment by NLP algorithms (Fig. 6.1).

(b) HazOp Ontology

The HazOp ontology defines, in a generic form, the concept of a study node, its elements and the relationships between them. These are types of study node, description, guidewords, deviations, causes, consequences, risk level, safeguards and recommendation. The HazOp ontology was developed using OWL DL language and consists of 17 classes, 23 object properties and 43 restrictions. Figure 6.3 presents a schematic view of the structure of the HazOp ontology. It has two important roles: (1) helping to identify potential hazards during study nodes recommendation since its specification clearly defines which type of domain concept could be a study node and (2) validation of the structure of the HazOp information before it is stored in the case library during case retention. A HazOp study node must be one of the types defined in the HazOp ontology.

(c) Ontology Library

The ontology library is a repository of domain ontologies. The domain ontologies (.owl/.rdf) could be those that have been developed for the purpose of

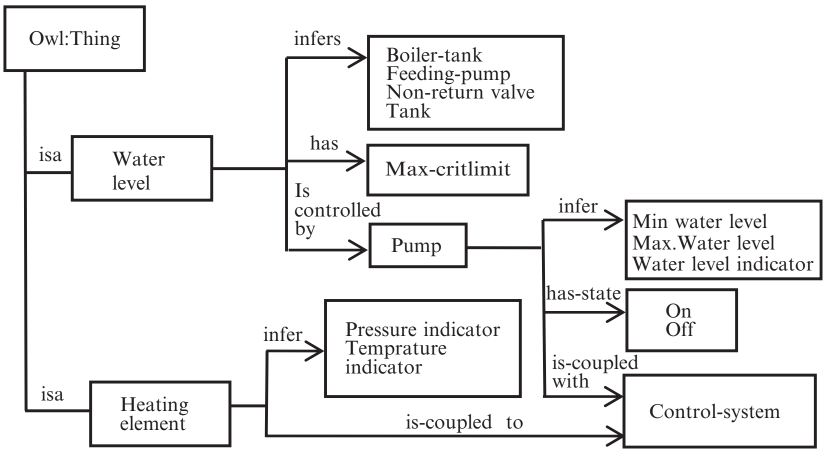


Fig. 6.2 A view of a part of the steam boiler ontology

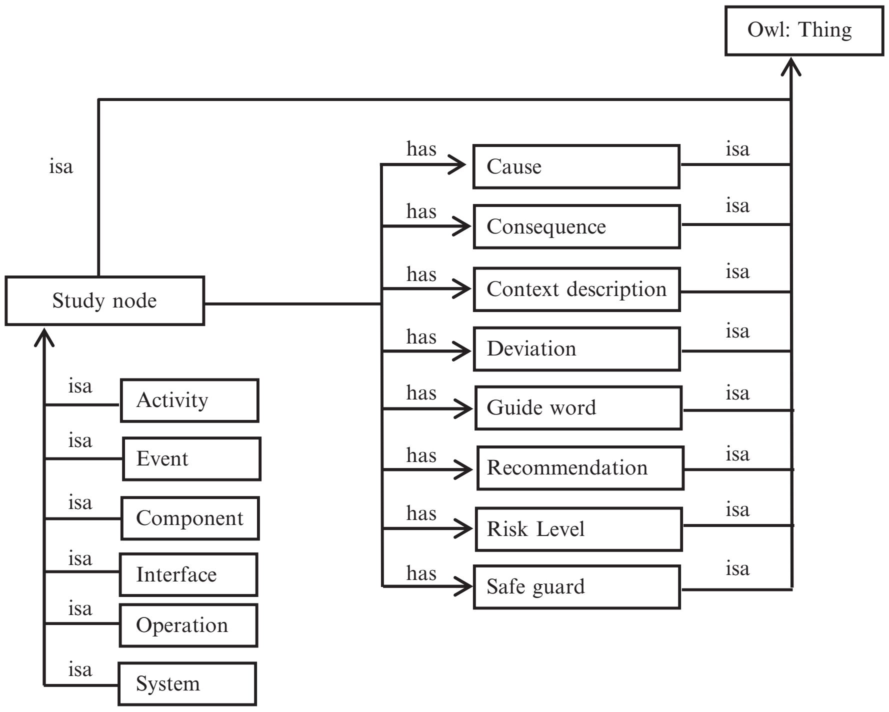


Fig. 6.3 A view of the classes and restrictions in the HAZOP ontology

safety analysis or an existing ontology that is based on domain-specific safety standards. The domain ontology consists of all the terms in the domain and the set of relationships between terms in the domain. The domain ontology plays two roles: (1) identification of valid domain concepts that are contained in requirements document and (2) ensuring that standardised terms used in describing HazOp information during knowledge (case) retention agree with the established vocabulary of the domain. As an example, a view of part of the steam boiler ontology which describes the concepts of the steam boiler system and the interrelationships between the concepts is shown in Fig. 6.2. The ontology library, the HazOp ontology and the case library jointly constitute the knowledge model of the framework.

(d) NL Processor

The NL processor component facilitates the processing of natural language and boilerplate requirements during the process of automatic recommendation of HazOp study nodes. The core natural language processing operations implemented in the architecture are:

- *Tokenisation*: Splitting of requirements statements (sentences) into word parts.
- *Parts of speech tagging*: Classification of tokens (words) in requirements statements into parts of speech such as noun, verb, adjective and pronoun.
- *Pronominal anaphora resolution*: The process of identifying pronouns (anaphors) which have noun phrases as antecedents in requirements statements. This is essential in associating sentences that refer to the same requirement.
- *Lexical parsing*: Creating the syntax tree that represents the grammatical structure of requirements statements, in order to determine phrases, subjects, objects and predicates.

The stanford NLP toolkit[3] for natural language processing was used to implement all NLP operations.

(e) Study Node Recommendation

The procedure for automatic study node recommendation is based on a heuristic algorithm that is derived from basic knowledge of HazOp. Study node generation is not intended to replace human capability but rather to create a credible starting point for early hazard identification and to alleviate the amount human effort involved. The algorithm searches for potential study nodes in two ways:

- *Requirements level (RL)*: A requirement statement is considered a candidate if the following criteria are satisfied: (1) the requirement statement contains an *action-entity* pair such as "open valve", "close valve", "start pump" or "stop pump" (*action* and *entity* may not necessarily follow each other in a sentence); (2) the *action* must be an instance of a generic HazOp action word (such as: stop, close, open, send, reset, cut, receive, start or their synonyms) or one of a set of

[3] http://nlp.stanford.edu/software/lex-parser.shtml

user specified keywords, while *entity* is a valid concept in the domain ontology; and (3) the entity identified in requirement statement belongs to one of the predefined study node types (components, system, etc.) as described in the HazOp ontology.

- *Component level(CL)*: A term (word) contained in a requirement statement is considered a candidate study node if the following criteria are satisfied: (1) the term is a valid concept in the domain ontology; (2) there exists at least one axiom that pertains to the term in the domain ontology which indicates that it could be a study node (In other words, it is one of several types of study nodes as defined by the HazOp ontology) and (3) the term has failure modes or guidewords defined on it (such as stuck, omission, commission) in the domain ontology. At the CL level, terms that satisfy the criteria (1) and (3), (1), (2) and (3) or (1) and (3) are considered to be candidates. However, a term is ignored if it is same as, equivalent to or a subclass of another term that has been selected as a potential study node.

6.4.2 Knowledge Reuse

This section describes the parts of the KROSA architecture that deals with knowledge reuse and also report generation.

(f) CBR Module

The CBR component facilitates the knowledge reuse capability of the framework. It emulates the typical workflow of the CBR life cycle which is retrieve, reuse, revise and retain [8, 24]. Retrieval by the CBR module is performed by displaying a ranked list of cases similar to a target case. Two types of reuse are supported: (1) total reuse – all parts of a case are reused for a new case, and (2) partial reuse – only parts of an existing case are reused in a target case. Revision can be effected by the HazOp expert by making modifications to the selected case to suit the new target case. Retention is done through storage of study node information into the case library. The case library is implemented as a MySQL database management system (DBMS) in order to leverage its inherent capabilities for effective case organisation, case indexing, case storage and case retrieval.

(g) Report Generation

This module enables the generation of HazOp reports based on query posed by the user. HazOp reports are queried based on date and the HazOp id.

6.4.3 Case Model and Case Similarity

The case model is an abstraction of the way HazOp information is represented in the framework. A HazOp case encapsulates information attributes such as name of a study node (unique), context description and set of applicable guidewords, deviations, causes, consequences, risk levels, safeguards and recommendations. The case model is partitioned into a problem part and a solution part. The three elements of a HazOp case model that constitute the problem part are contextual description, study node type and the set of guidewords; the remaining elements of the case model make up the solution part.

At the instance of a new (target) case, an algorithm is used to compute the similarity between the problem parts of the new case and all existing relevant cases in the case library to determine suitable candidates for retrieval. The solution part of a chosen retrieved case is then used verbatim or revised as the solution part of the target case. There are several candidate similarity algorithms that can be used for case retrieval depending on the value of attributes of data elements [25]. The similarity algorithm used for comparing cases is based on the degree of intersection between two attributes of a case, which are the set of contextual descriptions and the set of guidewords, while the type of study node is used to determine relevant cases. Similarity between an attribute of the new case U and a corresponding attribute of an existing case V is determined by computing the metric:

$$Sim(U,V) = \frac{|U \cap V|}{|U|} \tag{6.1}$$

where

$$U \cap V = \{x : x \in U and x \in V\}$$

Case Similarity: Finally, the similarity between two cases is computed by using the weighted sum of the individual similarity metrics, where w_i denotes the weight assigned to the ith attribute of a case. This is given as [26]:

$$Sim\_final = w_1 sim_{(context)} + w_2 sim_{2(guidewords)} \tag{6.2}$$

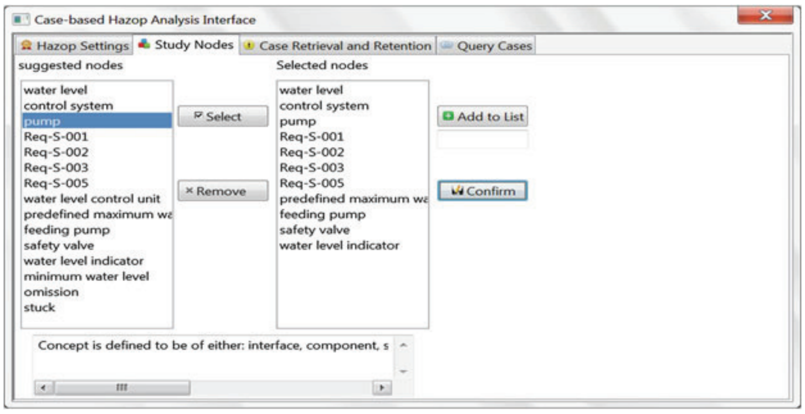
We have used equal weights (i.e., $w_1 = w_2 = 1$) since the parameters are considered as equally important.

6.5 Performing HazOp with KROSA

The process of using the KROSA tool for HazOp is as follows:

Step 1: Preprocessing of source documents to get the requirements into MS Excel or text file format and devoid of graphics, images and tables.

Step 2: Select existing domain ontology or create a new one to be used for the HazOp.



suggested nodes

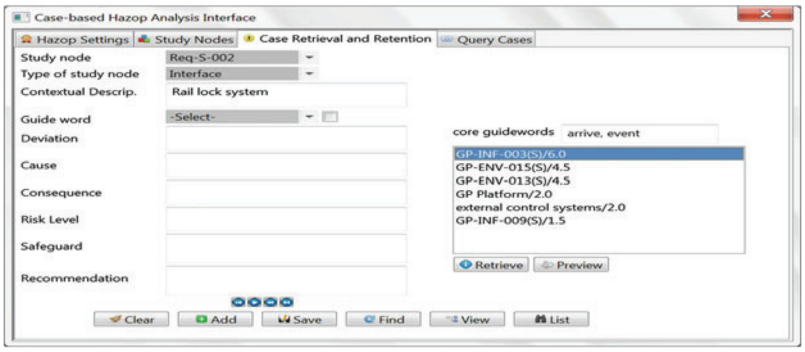
water level
control system
pump
Req-S-001
Req-S-002
Req-S-003
Req-S-005
water level control unit
predefined maximum wa
feeding pump
safety valve
water level indicator
minimum water level
omission
stuck

Selected nodes

water level
control system
pump
Req-S-001
Req-S-002
Req-S-003
Req-S-005
predefined maximum wa
feeding pump
safety valve
water level indicator

Concept is defined to be of either: interface, component, s

Fig. 6.4 A view of recommended study nodes by the KROSA tool



Study node	Req-S-002
Type of study node	Interface
Contextual Descrip.	Rail lock system
Guide word	-Select-
Deviation	
Cause	
Consequence	
Risk Level	
Safeguard	
Recommendation	

core guidewords arrive, event

GP-INF-003(S)/6.0
GP-ENV-015(S)/4.5
GP-ENV-013(S)/4.5
GP Platform/2.0
external control systems/2.0
GP-INF-009(S)/1.5

Fig. 6.5 A view of ranked list of similar cased retrieved by the tool

Step 3: Import requirement documents and domain ontology into the KROSA environment.

Step 4: Supply the set of keywords that best describe the focus of the HazOp.

Step 5: Obtain recommended study nodes from KROSA.

Step 6: Expert approves a set of study nodes for the HazOp by selecting from or adding to the recommendations by KROSA.

Step 7: For each approved study node, expert leverages KROSA's case retrieval, reuse and retention features to generate information for specific study nodes. By doing so, the user attempts to save some effort by using content of the reuse repository to provide information for new study nodes. Figures 6.4 and 6.5 are snapshots of the interfaces for study node recommendation and case retrieval for reuse in the KROSA tool, respectively.

6.6 Evaluation

We have developed the KROSA[4] (knowledge reuse-oriented safety analysis) tool, a domain-independent CBR platform for ontology-supported HazOp that is based on the Eclipse plug-in architecture. In following subsections, we discuss how the KROSA tool can be integrated into the HazOp process and subsequently describe the procedure used for its evaluation.

6.6.1 Evaluation Procedure

KROSA has been subjected to two kinds of evaluation: first, an in-house simulation experiment to assess the quality of its recommendation of study nodes, using requirements specifications obtained from ABB Norway, one of our partners in the CESAR project. Second, we performed a field assessment where industry experts from ABB Norway assessed the usability of KROSA for an industrial HazOp process. The objectives of the field evaluation were threefold: (1) to assess the consistency of the outcome of the tool as judged by the human experts, (2) to assess the potential of the tool to enable reuse-oriented HazOp and (3) to determine its usefulness as a support tool for safety analysis. Also, we wanted to identify areas of possible improvement of the tool.

6.6.1.1 Simulation Experiment

In the simulation experiment, we worked with three sets of requirements: (1) rail lock system, (2) steam boiler control system and (3) adaptive cruise control (ACC) system. Three ontologies used for the experiment are *rail lock system ontology*, *steam boiler ontology* and *ACC ontology*. Two of the ontologies (steam boiler and ACC ontology) had existed prior to KROSA, having been used to support previous ontology-based research project in CESAR [27]. These two ontologies have a fairly wide circulation among CESAR partners. The rail lock system ontology was created for this experiment, based on information obtained from the specification of the GP rail lock system. The three ontologies have the common characteristics that they were developed to be usable for safety analysis in addition to other uses. This is because (1) safety relevant terms were used to describe ontological concepts, e.g., object properties such as *isComponent*, *isConcept*, *isFailuremode* and *isInterface* exist in the ontologies; (2) the semantic description of components included the definition of generic failure modes such as *stuck*, *omission* and *commission*. The simulation experiment compared recommendations from KROSA with those obtained from four safety experts (researchers) for the same

[4] KROSA tool can be downloaded at https://www.idi.ntnu.no/~wande/Krosa-user-guide.htm

set of requirements. We then computed the recall and precision scores for KROSA relative to the recommendations made by each of the four safety experts that participated in the experiment (see Eqs. 6.3 and 6.4):

$$precision = \frac{|\{Expert.recomm\}| \cap |\{KROSA.recomm\}|}{|\{KROSA.recomm\}|} \tag{6.3}$$

$$recall = \frac{|\{Expert.recomm\}| \cap |\{KROSA.recomm\}|}{|\{Expert.recomm\}|} \tag{6.4}$$

6.6.1.2 Expert Assessment

For the field assessment, the direct method of expert systems evaluation [28, 29] was used. This method entails making qualified human experts to use a system for solving a simple benchmark problem; thereafter based on their experience, the human expert answers a set of questions about the system. The questions are quantitative and based on a 0 (completely false) to 5 (very true) numerical scales. A metric called "satisfaction level" that ranges from 0 (least satisfied user) to 5 (most satisfied user) is then computed based on the data obtained from all participants. The satisfaction level is a measure of the likelihood of the system to satisfy a prospective user.

The questions, the objective of each question and the weight associated with each question (which all the participants agreed on) are as follows:

1. *Sufficient information is provided for guidance and orientation of evaluators prior to conducting the experiment* (orientation) – (2).
2. *The KROSA tool reaches a conclusion similar to that of a human expert* (correctness of result) – (2).
3. *Does the KROSA tool provide reasonable justification for its conclusion?* (correctness of result) – (2).
4. *The KROSA tool is accurate in its suggestions of study nodes* (accuracy of result) – (2).
5. *The result is complete. The user does not need to do additional work to get a usable result* (accuracy of result) – (2).
6. *Does the result of the system change if changes are made to the system parameters?* (sensitivity) – (1).
7. *The overall usability of the KROSA tool is satisfactory* (confidence) – (1).
8. *The KROSA tool gives useful conclusions* (confidence) – (2).
9. *The KROSA tool adequately supports reuse of knowledge for HazOp* (support for reuse) – (2).
10. *The KROSA tool improves as data, or experience is inserted* (support for reuse) – (1).

11. *The limitations of the KROSA tool can be detected at this point in time* (limitation) – (1).
12. *There are still many limitations to make the KROSA tool usable* (limitation) – (1).

An evaluator gives a score between 0 and 5 per question. From the scores, a weighted score for the satisfaction level per evaluator can be calculated using the metric below:

$$\mathrm{Re}\ sult = \sum_{k=1}^{n} (weight^{*} scorevalue) / \sum_{k=1}^{n} weight \tag{6.5}$$

where n is the number of questions.

A one-day orientation workshop on how to use the tool was conducted for all participants, after which they had one full week to interact with the tool. The expert participants also had a detailed user manual as further guide for using the tool.

6.7 Evaluation Results

In this section, we give an overview of results from the two evaluations carried out.

6.7.1 Simulation

Table 6.2 shows the recall and precision scores computed for KROSA relative to the four safety experts' (E1–E4) recommendations. Although the experts differed in their recommendations, confirming the subjective nature of HazOp, there exist significant agreements between study nodes recommended by KROSA and experts at the requirements level. At the component level (CL), there was a greater degree of agreement because the opinions of the safety experts generally agree that all components and interfaces between components and systems should be study nodes as recommended by KROSA. Since the experts were generally not very specific in their recommendations at the CL, recommendations at CL were not considered when arriving at the values in Table 6.2. The result – precision[5] and recall[6] values – shown in Table 6.2 is an improved version of the one reported in [27] since we have had more time to improve on the quality of the domain ontologies.

Our observation from the simulation experiment (see Figs. 6.6 and 6.7) is that the performance of the KROSA tool depends significantly on the quality of the

[5] Precision – percentage of suggested hazards that are relevant compared to expert's recommendation.

[6] Recall – percentage of relevant hazards suggested by tool compared to expert's recommendation.

Table 6.2 Showing recall and precision values of KROSA

Recall	E1	E2	E3	E4
Steam boiler system (HazOp on water level)	0.57	0.67	0.75	0.60
ACC system (HazOp on speed control)	0.50	0.67	0.71	0.60
Rail lock system (HazOp on communication)	0.54	0.78	0.71	0.60
Precision	**E1**	**E2**	**E3**	**E4**
Steam boiler system (HazOp on water level)	0.67	0.67	0.50	0.50
ACC system (HazOp on speed control)	0.80	0.80	1.0	0.6
Rail lock system (HazOp on communication)	0.78	0.78	0.56	0.33

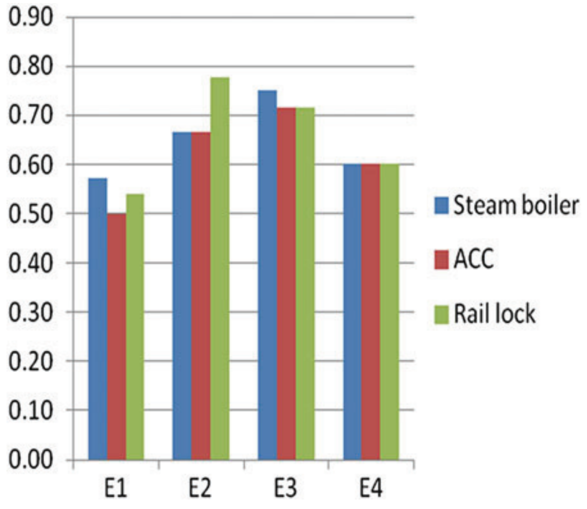


Fig. 6.6 Recall metric for KROSA

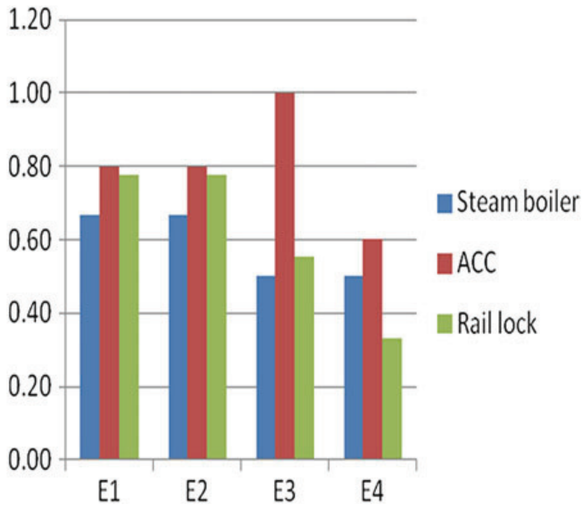


Fig. 6.7 Precision metric for KROSA

domain ontology, even though the input of highly relevant keywords can enhance the appropriateness of the recommended study nodes. Specific ontology qualities are considered most crucial here, which are [30] (1) *syntactic quality* – the measure of the correctness of terms in the ontology and the richness of syntax used to describe terms in the ontology, (2) *semantic quality* – the measure how well the meaning of terms is defined in the ontology and (3) *pragmatic quality* – the measure of the how well it covers the scope of a domain judged by the number of classes and properties it contains and how accurate and relevant the information is that it provides. A domain ontology that contains a large number of concepts that are credible and are richly described with axioms will be more suitable for KROSA in the task of study node recommendation. Initially, we noticed that the KROSA tool had a relatively lower precision for HazOp of the steam boiler system compared to its performance in the HazOp for the ACC and rail lock systems. The reason for this was that the steam boiler ontology has a lower semantic quality than the ACC ontology and rail lock system ontology. After we improved on the quality of description of concepts and interrelationships between concepts of the steam boiler ontology, we obtained better results. This is not difficult to comprehend since the domain ontology provides the knowledge base from which inferences are made by the KROSA tool when determining what could be a potential system hazard (study node). Thus, we conclude that the overall quality of the domain ontology affects the performance of KROSA tool significantly, as it determines the extent to which inferences can be made for identification of study nodes.

6.7.2 Expert Evaluation

Each of the three industry experts that took part in the assessment returned an evaluation report from which we computed a mean weighted score of 3.27 out of 5 for the KROSA tool in relation to the evaluation objectives of the field assessment. The tool obtained its highest mean score ratings in the aspects of *support for reuse* (4.08), *sensitivity* (3.67), *confidence* (3.25) *and accuracy of result* (3.25), while the lowest mean score ratings were in the aspects of: limitations (3.0) and *correctness of result* (2.7). These mean score ratings reveal the perception of the experts in terms of the strengths and weaknesses of the current version of the tool. The experts also submitted a detailed report on desired improvements needed to make the tool more usable. Key aspects mentioned as needing improvement were (1) the possibility of providing some form of guidance to users in the selection of the most appropriate keywords for study node recommendation and (2) the need to provide some form of traceability links between cases that have inherited some parts from old cases through reuse. The experts were unanimous in confirming that the tool will be a valuable support for the conduct of HazOp, with the potential to alleviate the complexity of the HazOp process by enabling reuse of experience.

The experts agreed that the existence of a domain ontology and a case library where previous knowledge is stored in a structured format would help to resolve some of the existing difficulties associated with searching, update and interoperability of knowledge during HazOp. They expressed preference for the adoption of KROSA as a support tool for HazOp over the current scenario where MS Excel software is the main tool support for their safety analysis.

6.7.3 Threats to Validity

Our short discussion on the validity of the preliminary evaluation will be based on the categories defined by Wohlin et al. in [31]. We consider each threat before giving a summary of validity of our results.

Conclusion Validity: In order to ensure reliable treatment, all participants were provided with an introduction and instructions for the experiment prior to the experiment. Also, we used standard measures – recall and precision to assess recommendations by the tool in order to avoid misunderstanding or misinterpretation of the results. Ordinarily using four participants in the experiment will translate to low statistical power, but for highly technical domain like HazOp and a preliminary evaluation, we consider this to be sufficient for a first trial.

Internal Validity: A key requirement is that participants have sufficient experience or knowledge of the domain. The participants had minimum master-level education in the area of systems safety. They were also provided with detailed instructions of what should be done. Therefore, there were no factors other than the treatment that influenced the outcome of the experiment.

Construct Validity: In order to ensure a realistic experiment, all participants had the same instruction for the experiment. Also, they performed exactly the same task which is to identify hazards (study nodes). Hence, the results obtained from participants depend only on this task (one single variable), which eliminates any mono-method bias effect.

External Validity: The key issue here is whether we can generalise our results from the preliminary evaluation to the system safety industry. For the simulation experiment, we used four expert researchers all affiliated with NTNU, while the industrial assessment was done by three safety experts at ABB Norway. A concern could be that possibly there would have been different results if the evaluations had been performed with a bigger group of participants with more diverse background, not only in terms of coming from different institutions and countries but also with more different educational backgrounds and covering a wider spectrum of safety-critical domains than could be achieved with only seven persons. The involved persons mainly had experience in safety analysis in the following domains: railway, automotive and industrial automation, and it is impossible to know if the tool would have been found equally promising by experts from other domains, such as nuclear power, medical technology and aviation. Our mitigation to this threat is to try to avoid including any domain-specific limitations in our general approach, but this does not entirely remove the threat. So, while we currently see no reason why the approach should not also be usable in other companies and other safety-critical domains, an interesting point for further research is to have a wider group of experts to try out the tool.

Hence, we cannot foresee any serious threats to validity for our conclusions on the simulation experiment performed. Also, the feedback for industry experts proved that the KROSA tool has sufficient merit for application in an industrial setting.

6.8 Related Work

Previously a number of attempts to solve some of the problems of HazOp analysis have been reported in the literature [1, 3, 7]. A significant number of HazOp expert systems and HazOp system prototypes have been reported in [4]. These include HazOpEX, Batch HazOpExpert, HazOp Diagraph Model (HDG), STOPHAZ, OptHazOp, EXPERTOP, HazOpTool and COMHazOp. A common trend for all of these attempts is that their implementation and application were focussed on the chemical process industry (CPI), the domain where HazOp originated. Also, they were essentially rule-based expert systems and were not designed to facilitate the reuse of experience [4]. Relatively few other automated tools for HazOp in other domains have been reported in the literature [4]. This situation possibly reveals the fact that the HazOp procedure in most cases is done manually but aided by the use of spreadsheet software packages such as MS Excel and Lotus 1-2-3 in many application domains.

It is only recently that case-based reasoning was introduced into HazOp and few efforts have been reported so far. Sahar et al. in [6] presents a report on development of a HazOp analysis management system with dynamic visual model aid. The system is based entirely on CBR with no ontology support for HazOp. In [7], a case-based expert system for automated HazOp analysis called PHASUITE was developed. The PHASUITE system caters to the modification of existing HazOp models and creation of new ones based on the knowledge in existing models. It is also equipped with diagnostic reasoning capability and is suitable mainly for process generic HazOp. It makes use of a suite of informally specified ontologies. PHASUITE is specialised for application in the chemical industry domain. The PetroHazOp [1] has specific application for the chemical domain and was developed to cater to both process generic and non-process generic HazOp. The system uses an integration of CBR and ontology for the automation of both process generic and non-process generic HazOp procedures.

The PetroHazOp [1] and PHASUITE [7] systems are the ones most related to our work since they are based on integration of CBR and ontology. However, none of them have the capability for HazId based on requirements nor are they designed to have any bearing or relevance to requirements engineering as conceived by our approach. Additionally, unlike the two aforementioned tools that are specialised for the chemical industry domain, our approach is a generic one that can be adapted to support several types (process, software, human or procedure) of HazOp analysis in different application domains, given the existence a relevant domain ontology. Hence, the novelty of our approach is the attempt to enable early identification of systems hazards right from the requirements engineering phase of system development and the reuse of experience in order to reduce the amount of resources needed for HazOp. The core idea of this chapter has been reported in [27] in abridged form.

6.9 Conclusion

This work offers support for knowledge management in systems engineering at two levels. Firstly, at the level of requirements, it facilitates the exploitation of knowledge contained in requirements documentation for early identification of potential system hazards. The novelty of this is the provision of tool-based support for safety analysis at an earlier phase of system development as compared to previous efforts that focus only on the design phase. Secondly, our approach facilitates the reuse of experience in the conduct of HazOp so that previously documented HazOp knowledge can be leveraged for reduced effort in new projects.

Specifically, we have provided a tool that can creditably assist, but not replace the human expert in the conduct of HazOp analysis so as to attain reduction in effort needed. Considering the fact that HazId is a highly creative process that depends on the experience and skill of the human domain expert, the KROSA tool would be vital as a good starting point. Also, from the results of the evaluation, KROSA has demonstrated a good potential for application in an industrial context. The tool would particularly be helpful in situations where highly skilled or experienced HazOp experts are not available, by enabling a platform whereby previously documented cases can be reused in new scenarios by a less-experienced HazOp team.

In further work, we intend to realise the objective of an extensive semantic framework for safety analysis by extending the features of KROSA to support FMEA. We will also investigate the prospects of providing diagnostic reasoning over potential hazards in order to facilitate a more elaborate automated safety analysis. In addition, we aim to conduct more extensive industrial case studies on safety analysis of systems and product lines using the tool and to report our findings subsequently.

Acknowledgements We appreciate the contributions of the staff of system safety research division ABB Norway in conducting trial evaluation of the KROSA tool.

References

1. Zhao J, Cui L, Zhao L, Qui T, Chen B (2009) Learning HAZOP expert system by case-based reasoning and ontology. Comp Chem Eng 33(1):371–378
2. Dittman L, Rademacher T, Zelewski S (2004) Performing FMEA using ontologies. In: The 18th international workshop on qualitative reasoning. North-western University, Evanston

3. Smith S, Harrison M (2005) Measuring reuse in hazard analysis. Reliab Eng Syst Safe 89(1):93–104
4. Dunjo J, Fthenakis V, Vilchez J, Arnaldos J (2010) Hazard and operability analysis: a literature review. J Hazard Mater 173(1–3):19–32
5. Mokos K, Meditskos G, Katsaros P, Bassiliades N, Vasiliades V (2010) Ontology-based model driven engineering for safety verification. In: Proceedings of 36th EUROMICRO conference on software engineering and advanced applications, Lille, pp 47–54
6. Sahar B, Ardi S, Kazuhiko S, Yoshiomi M, Hirotsugu M (2010) HAZOP management system with dynamic visual model aid. Am J Appl Sci 7(7):943–948
7. Zhao C, Bhushan M, Venkatasubramanian V (2005) PHASUITE: an automated HazOp analysis tool for chemical processes Part I: knowledge engineering framework. Proc Safe Environ Prot 83(B6):509–532
8. Kolodner J (1992) An introduction to case-based reasoning. Artif Intell Rev 6(1):3–34
9. Gruber T (1993) A translation approach to portable ontologies. Knowl Acquis 5(2):199–220
10. Jurafsky D, Martin JH (2008) Speech and language processing: an introduction to natural language processing. Speech recognition and computational linguistics, 2nd edn. Prentice-Hall, Upper Saddle River
11. Redmill F, Chudleigh M, Catmur J (1999) System safety: HazOp and software HazOp. Wiley, New York
12. Shepperd M (2003) Case-based reasoning and software engineering. In: Aurum A (ed) Managing software engineering knowledge. Springer, Berlin
13. Kotis K, Vouros G (2005) Human-centered ontology engineering: the HCOME methodology. Knowl Inform Syst 10(1):109–131
14. Happel HJ, Maalej W, Seedorf S (2010) Applications of ontologies in collaborative software development. In: Mistrík I et al (eds) Collaborative software engineering. Springer, London
15. Lohmann S, Heim P, Auer S, Dietzold S, Riechert, T (2008) Semantifying requirements engineering – the softWiki approach. In: I-SEMANTICS, Graz, pp 182–185
16. Dzung DV, Ohnishi A (2009) Ontology-based reasoning in requirements elicitation. In: Proceedings of seventh international conference on software engineering and formal methods, Hanoi, pp 263–272
17. Lee Y, Zhao W (2006) An ontology-based approach for domain requirements elicitation and analysis. In: Proceedings of the first international multi-symposiums on computer and computational sciences, Hanzhou, Zhejiang
18. Shibaoka M, Kaiya H, Saeki M (2007) GOORE: Goal-oriented and ontology driven requirements elicitation method, ER workshops 2007, Auckland, Lecture notes in computer sciences 4802, Springer, Heidelberg, pp 225–234
19. Maiden N, Sutcliffe A (1993) Case-based reasoning in software engineering. In: IEEE colloquium on case-based reasoning, London, pp 2/1–2/3
20. Jani M (2010) Applying case-based reasoning to software requirements specifications quality analysis system. In: Proceeding of 2nd international conference on software engineering and data mining (SEDM), IEEE Press, Chengdu, pp 140–144
21. Daramola O, Moser T, Sindre G, Biffl S (2012) Managing implicit requirements using semantic case-based reasoning. In Regnell B, Damian D (eds) REFSQ 2012, Lecture notes in computer sciences 7195, Springer-Verlag, Berlin/Heidelberg, pp. 172–178
22. Hull E, Jackson K, Dick K (2004) Requirements engineering. Springer, London
23. Stålhane T, Omoronyia I, Reichenbach F (2010) Ontology-guided requirements and safety analysis. In: Proceedings of international conference of emerging technology for automation, Tampere
24. Aamodt A, Plaza E (1994) Case-based reasoning: foundational issues methodological variations, and system approaches. Artif Intell Commun 7(1):39–59
25. Pedersen T, Pakhomov S, Patwardhan S, Chute C (2007) Measure of semantic similarity and relatedness in biomedical domain. J Biomed Inform 40(3):288–299

26. Ferguson A, Bridge D (2000) Generalised weighting: a generic combining form for similarity metrics. In: Proceedings of the 11th Irish conference on artificial intelligence & cognitive science (AICS’2000), pp 169–179
27. Daramola O, Stålhane T, Sindre G, Omoronyia I (2011) Enabling hazard identification from requirements and reuse-oriented HAZOP analysis. In: Proceeding of 4th international workshop on managing requirements knowledge, IEEE Press, pp 3–11
28. Daramola O, Oladipupo O, Musa A (2010) A fuzzy expert system tool for personnel recruitment. Int J Bus Inform Syst 6(4):444–462
29. Salim M, Villavicencio A, Timmerman M (2002) A method for evaluating expert system shells for classroom instruction. J Indust Technol 19(1):Nov 2002–Jan 2003
30. Burton-Jones A, Storey V, Sugumaran V, Ahluwalia P (2005) A semiotic metrics suite for assessing the quality of ontologies. Data Knowl Eng 55:84–102
31. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2000) Experimentation in software engineering: an introduction. Kluwer, Norwell