Identifying Risks for Collaborative Systems during Requirements Engineering: An Ontology-Based Approach

Kirthy Kolluri, Robert Ahn, Julie Rauer, Lawrence Chung
Department of Computer Science
The University of Texas at Dallas
Richardson, TX, USA
{kirthy.kolluri, robert.sungsoo.ahn, julie.rauer, chung}@utdallas.edu

Tom Hill
Fellows Consulting Group
Dallas, TX, USA
{tom}@fellowsconsultinggroup.com

Abstract- A risk is an undesirable event that can result in mishaps if not identified early on during requirements engineering adequately. However, identifying risks can be challenging, and requirements engineers may not always be aware if risks are ignored. In this paper, we present *Murphy* – a framework for performing risk analysis. Murphy adopts the *Reference Model*, in which requirements are supposed to be met not by the projected software system behavior alone but through collaboration between the system and events occurring in its environment, hence the term *Collaborative System*. Murphy provides risk analysis facilities that include an activity-oriented ontology for carrying out risk analysis by systematically identifying risky activities in the system and in the environment, thereby obtaining a *Risk Analysis Graph (RAG)* and towards devising risk mitigation strategies later. In order to see both the strengths and weaknesses of Murphy, we experimented on developing a smartphone app involving a group of Ph.D. and senior-level graduate students – one group using Murphy and the other not using Murphy. Our observation, we feel, shows that the risks identified by the group using Murphy were able to identify more critical risks and those risks were comprehensive and relevant as. well. The results also showed that incorporating risk mitigation strategies for the risks identified can indeed help avoid them to some extent.

Keywords- ***Risk, Risk Identification, Ontology, Collaborative systems, Requirements Engineering, Reference Model (WRSPM Model)***

I. INTRODUCTION

Risk is a situation or event where humans themselves can be put at stake [13], is a phenomenon faced or caused by erroneous functionality/behavior of software, hardware, or human(s). Collaborative systems emphasize that requirements are satisfied by the collaboration between the user and the events in its environment. For example, in building our smartphone app, *Theia*[1], for helping blind people navigate indoors, it may not be too evident for the requirements engineer to identify that the "blind person may not be able to walk in a straight line." It can be challenging to determine the possibility of the smartphone app giving wrong instruction to the blind person, the camera not turning on even when the smartphone application is turned on, etc. Based on these examples, it is evident that risks may arise due to certain environmental events (domain) or erroneous system behavior.

When considering collaborative systems such as Theia, an agent (e.g., person, software, or hardware) must perform a set of activities to fulfill the requirement. Every action performed by the agent, or the software system is associated with one or more risks. What-if the smartphone app indicates the blind person to turn earlier or later after walking ten steps? Or what if the user ignores the instructions and fails to turn at the right spot? The requirements engineers and software developers should address these kinds of risks before developing the actual application. This practice would help plan risk minimization and mitigation strategies.

Some attempts have been made to perform risk analysis and mitigation during requirements analysis [1]. The idea here is to identify risks systematically, devise risk mitigation strategies and implement those strategies to help avoid some risks. But how do we systematically identify these risks without omissions and commissions and develop risk mitigation strategies?

This paper proposes *Murphy*, a framework for performing risk identification and analysis using an activity-oriented and ontology-based approach for collaborative systems. Our previous work [19], which extends the Reference Model [6, 7] with risks to obtain the Augmented Reference Model, is extended by adding more rules for systematic risk identification. We also introduce a highly activity-oriented ontology, a domain-specific ontology, and constraints on agent's actions which can be used to come up with risks when violated.

We carried out experimentation in two phases – *Phase 1* involved using the Theia app developed without using Murphy before its development. *Phase 2* involved using Theia app developed using Murphy framework. Through this experimentation, we have observed that significant risks, such as walking in a straight line, the user's finger being slippery to tap the screen, background noise, etc., can be overlooked. We also observed that the devising and developing risk mitigation strategies can indeed help avoid the occurrence of the risk to some extent.

DOI reference number: 10.18293/SEKE2022-169

[1] Theia is the Greek goddess of sight

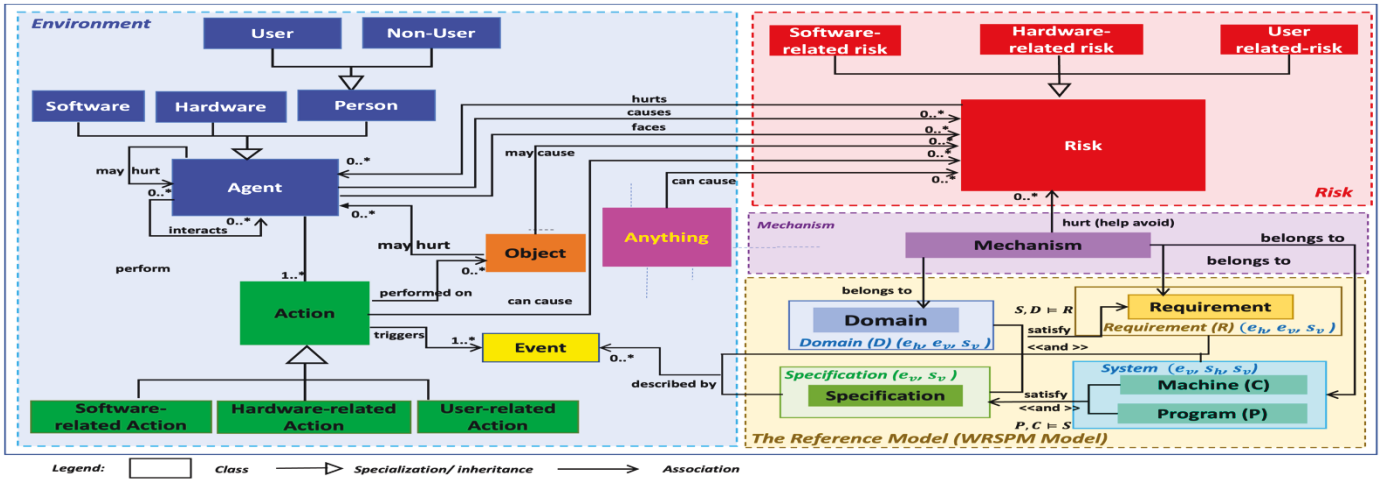

Figure 1: High-Level domain independent ontology of Murphy for risk identification and analysis

The main contributions of this paper are: proposing a risk analysis framework for capturing risks for various activit--ies performed by the Agent (Person, Software and Hardware) using multiple levels of ontologies during the requirement engineering phase. The framework suggests that risk mitigation strategies that must be implemented during the development of the application.

A scenario using the indoor navigation application, Theia, for helping blind people navigate indoors is used as the running example all through this paper, to evaluate the strengths and weaknesses of Murphy. Stevie is a blind student who wants to attend a class in room 3.415. He uses the smartphone application, Theia, to navigate from his current location to his class. He uses voice instruction to provide his destination to Theia.

Section II describes the related work. Section III describes Murphy framework using an ontology-based approach for risk identification. Section IV describes the experimentation and the observations, the discussion, and threats to validity. In the end, a summary of the paper is described, along with some future work in Section V.

II. Related Work

The distinctives of this paper include performing risk analysis for Collaborative systems in terms of the Augmented Reference Model, Risk Analysis Graph, and an Ontology-based approach, during requirements engineering.

Concerning risk analysis for collaborative systems, the CORAS framework [8], goal-risk framework [1, 10] and the Obstacle analysis technique [4, 5, 9, 11] are considered. All these frameworks consider non-functional requirements (goals) as their starting point and risks are eventually identified using different approaches, but we consider functional-requirements, specification, and domain assumptions as the starting point to perform risk identification and analysis.

In CORAS [8], risks are modeled and analyzed by asking questions, which are evaluated and treatments to those risks are identified. In the goal-risk framework [1, 10] goals, events and treatments are modeled in three layers, and they provide multi-object optimization; hence more queries related to risk are obtained qualitatively. Using the technique of obstacle analysis [4, 5, 9, 11], goals are decomposed into sub-goals, providing a set of rules including negation to identify the probability of risk occurrence quantitatively. Though our work has some similarities with [8, 1, 10, 4, 5, 9, 11] with regards to the approach of decomposing/refinements, and performing qualitative risk analysis, we complement the approaches used in [8, 4, 5, 9, 11] by performing systematic risk identification using an activity- oriented ontology. We also perform qualitative risk as discussed in [1] but we complement the analysis performed by using the Risk Analysis Graph, which provides a set of rules, using which the user can obtain and identify risks, prioritize them, and devise the corresponding risk mitigation strategies.

The Reference Model [6] emphasizes that the user requirements are satisfied not by the system alone but also by the system's collaboration with the events in its environment. Hence, we use the term Collaborative system for all the systems to which the Reference model is applicable. We adopt work involving the Reference Model [6, 7] and transform it into Augmented Reference Model [19].

In Requirements Engineering, a Fishbone diagram has been used to identify possible causes for a problem/risk [16]. This technique helps list out all the potential causes for a problem/risk. Fault Tree Analysis (FTA) is a top-down, deductive analysis that visually shows a failure path from top to bottom [17]. A Problem Inter-Dependency Graph (PIG) uses a (soft)problem technique to represent user's problem against the user's goals [18]. We build on the idea of finding the problems from Fishbone diagram [16], to use a top-down approach from FTA [17] and making refinements to identify risks from [18], but we also complement these techniques by proposing the use of a Risk Analysis Graph (RAG), that refines requirements, specification and domain using AND/OR refinements, and systematically generate risks using rules and using an activity-oriented ontology to identify the essential/critical risks.

The ontology of risk discussed in [12] explains its relationship with value. The ontology discussed in Requirements Modelling Language (RML) [14] address Requirements, Agent, action, etc. as the most important concepts. We adopt those basic building blocks from RML [14] and build the most essential part of our work - Risk, on top of it and tie the con-

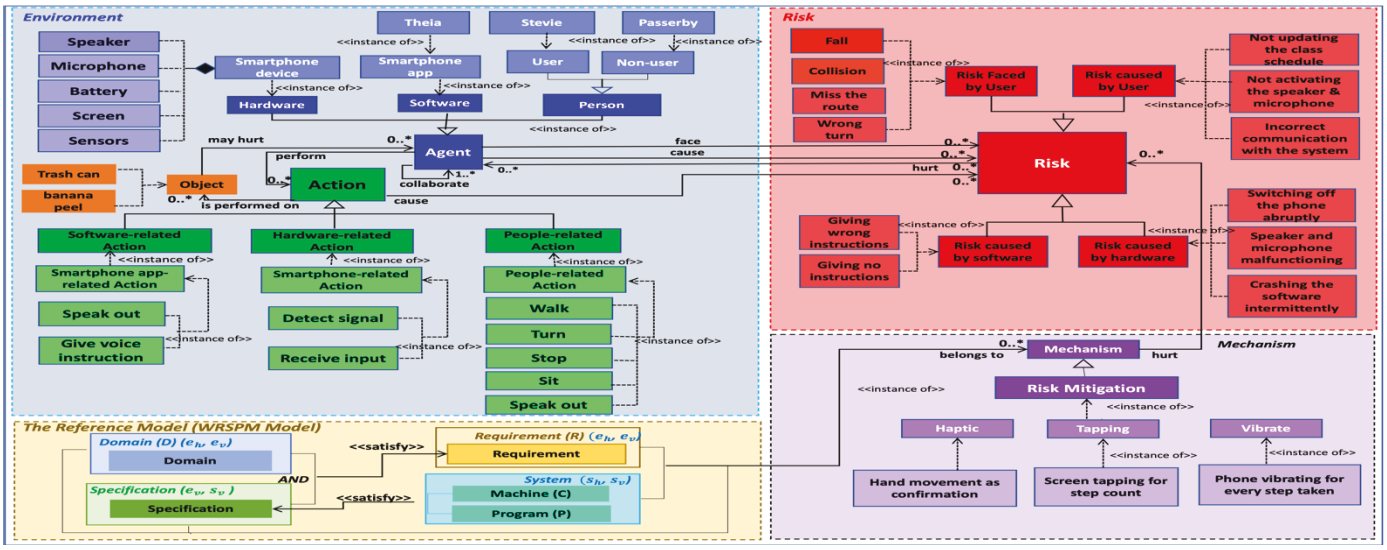

Figure 2: Domain specific activity-oriented ontology for smartphone application (Theia) domain

cept of Risk to Action and Agent. We also complement the approach discussed in [9, 12] by linking the ontology to actions and the risks that the user may face or cause.

III. MURPHY: AN ONTOLOGY-BASED FRAMEWORK FOR RISK IDENTIFICATION

The aim of our framework is to use an ontology-based approach to generate a Risk Analysis Graph using rules to identify risks, prioritize them and to find the appropriate risk mitigation techniques.

Ontology is the categories of essential individual concepts, relationship between the individual concepts and constraints on individual concepts and on the relationships between individual concepts. In this work, we present both high-level domain independent ontology and the domain specific class-level ontology with its diagrammatic convention as shown in Fig. 1, Fig. 2. We adopted several models and tightly integrated them into the ontology. Concepts for requirements engineering such as Requirement, Specification and Domain are adopted from Reference Model [6, 7] and the concepts such as Action (activities), Agent (entities) and Associations (assertions) from RML [14]. We further extend it by adding risk as shown in Fig. 1, which is one of the most essential concepts for this work alongside activities. Each concept is assigned a color and the same color is used for those concepts through the paper for traceability.

In detail, different types of *Agents* related to the domain such as Person, Software and Hardware are captured, the *Actions* pertaining to the agents such as Person action, Software action and Hardware action and the different types of *Risks* such as risks caused and faced by the person, risks caused by the software and risks caused by the hardware are captured. Capturing agents, their actions and the risks is vital because each agent performs a set of activities, and each activity is associated with a group of risks. In collaborative systems, agents interact and collaborate among themselves to fulfill the requirements. In addition, since the focus is on collaboration and collaborative approach, we capture all the critical concepts of the Reference Model [5] namely Requirement, Specification, Domain, Program and Machine in the ontology. Mechanism is captured to provide risk mitigation techniques for the risks discovered during risk identification and analysis process. Instances of the ontological concepts are represented in fig. 2 which is specific to Theia domain.

A. Omissions and Commissions*:* Using an ontology is one of the best ways to help ensure the completeness and comprehensiveness of the risk identification and analysis process. Identifying the most relevant and critical risks related to a domain help guarantee completeness, while identifying different kinds of risks is about comprehensiveness. Ontology helps avoid omissions and commissions of the most important/critical risks while performing risk analysis. Omissions are ignoring the most essential risks while commissions are identifying irrelevant or inconsistent risks. For e.g.,

R: When the person indicates a room number as the destination, the smartphone app shall ask the user to perform an action

By instantiating R using the class-level ontology shown in Fig. 2, it will be

r_1: When Stevie indicates his destination as room 3.415, Theia shall instruct Stevie to fly

The actions related to the agent (Person) as shown in Fig. 2 are walk, turn, stop, sit, speak out but we do not see fly as one of the actions related to Theia domain. Hence, this is an example of *commission*. Similarly, another instance for R can be:

r_2: *When Stevie indicates room 3.415 as his destination, Theia shall instruct Stevie to walk 10 steps forward.*

Stevie walking 10 steps forward can have so many risks associated with it. He may not walk forward at all but *backwards*, or he may not walk in a straight line but in a *zig-zag pattern*, or does not hold the camera facing forward but *downwards,* etc. All these are omission of substantial risks in relation to Theia domain. Stevie not hitting the brakes, Stevie not chang-

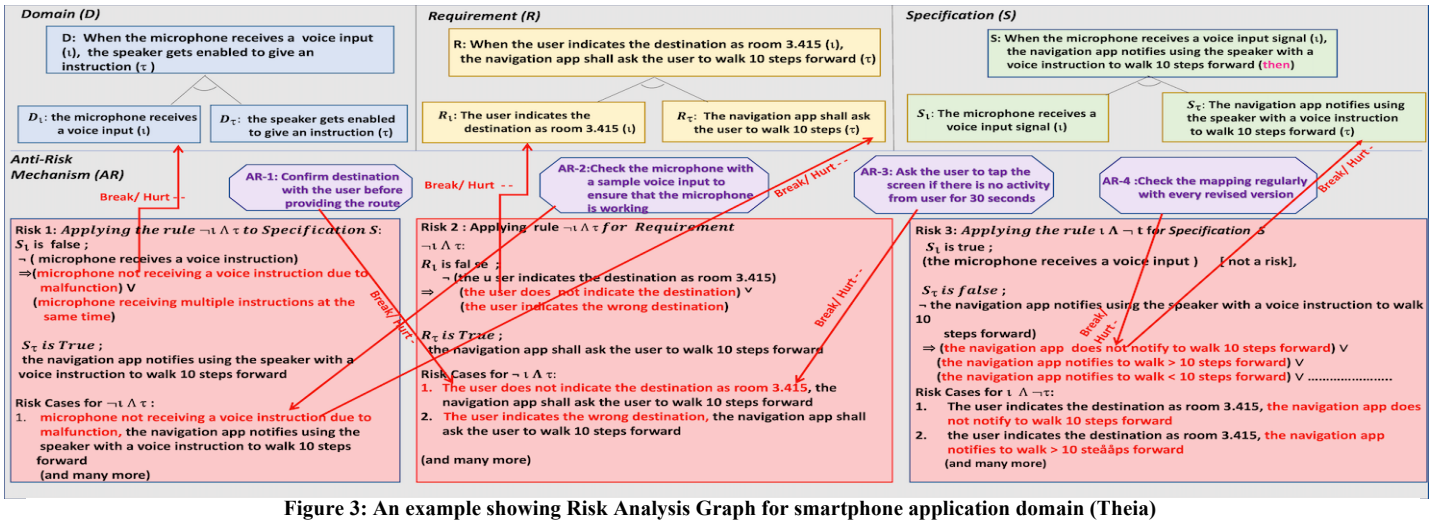

Figure 3: An example showing Risk Analysis Graph for smartphone application domain (Theia)

ing lanes, etc. are also risks but are not omissions related to Theia domain.

***B. Identifying risks using Constraints*:** Constraints are some restrictions that are placed on the ontological concepts and the relationships between them. Since, the ontology is highly activity-oriented, we place constraints and violation of these constraints is nothing but a risk. The constraints are specific to the domain and are related to the actions that the agent perform in the domain. For e.g., if we consider a constraint that the blind person must walk in a straight line when using Theia. The blind person walking in a zig-zag pattern can violate this constraint, which results in a risk.

***C. Generation of Risk Analysis Graph*:** In this paper, we generate Risk Analysis Graph, shown in Fig. 3, by the systematic generation of risks. For this, we use the Augmented Reference Model from our previous work [19] and extend it by adding multiple rules for extensive risk generation and devising risk mitigation strategies for the risks identified. As a part of the risk analysis process, we follow these steps to generate the RAG:

1) acquiring the requirement, specification, and domain,
2) decomposing the requirement, specification, and domain
3) applying rules to the decomposed requirement, specification, and domain to systematically obtain risks
4) prioritize the most important risks using ontology
5) devise risk mitigation strategies to the risks prioritized

Step 1 and 2: For this work, we assume that the requirement, specification, and domain are of the form $\iota \rightarrow \tau$. We use the antecedent (ι) part and the consequent (τ) part to identify risks by applying rules for systematic risk generation. We use a part of the initial process (Steps 1 and 2) discussed in our previous work [19] for generation of RAG.

Step 3: We have discussed some rules in our previous work [19], and we extend them by adding more rules for systematic risk generation. Rules are applied to the decomposed requirement, specification and domain that are obtained using the augmentation process explained in our previous work [19].

a) Rule 1: $\neg(\iota \rightarrow \tau)$ which is $\iota \wedge \neg\tau$

b) Rule 2: $\neg\iota \wedge \tau$: We consider the antecedent and the negation of the consequent joined by a logical AND

c) Rule 3: Negation of Contrapositive: Contrapositive is the reversal and negation of both i and t in $\iota \rightarrow \tau$. It is read as if not t then not i. We consider negation of the contrapositive [15], *represented as $\neg(\neg\tau \rightarrow \neg\iota)$*

In fig. 3, the first two red boxes show rule 2 in action and the last red box (towards right) shows rule 1 in action. We will discuss only rule 1 here (the last red box in fig 3) due to space limitation. Let us consider the specification S,

S: When the hardware receives a signal, the software notifies using the hardware to perform an action

To generate RAG, we perform step 1, i.e., acquiring s. We instantiate this S, using the ontology. By instantiating this specification S, we acquire:

s: When the microphone receives a voice input signal, Theia notifies using the speaker to walk 10 steps forward

We then perform step 2, i.e., decomposing s since we have an implies relation between ι and τ. After decomposing s, we obtain s_ι *and* s_τ.
s_ι: *the microphone receives a voice input signal*
s_τ: *Theia notifies using the speaker to walk 10 steps forward*

Now, we apply rule 1, $\iota \wedge \neg \tau$, to s_ι and s_τ. When rule 1 is applied to s, the consequent, s_τ, is negated since the consequent in $\iota \wedge \neg \tau$ has the negation.
The antecedent remains the same, and the relation between them is AND. By negating s_τ, we obtain:
(s_τ): ¬ (*Theia notifies using the speaker to walk 10 steps forward)*

There can be multiple risk cases associated with this negation.
(Theia does not notify to walk forward) OR
(Theia notifies to walk > 10 steps forward) OR
(Theia notifies to walk < 10 steps forward) OR
(Theia notifies to walk 10 steps forward and turn left) OR
(Theia notifies to walk < 10 steps forward and turn left)

When Theia must deliver a notification using the speaker, after calculating the route, there may be a set of risks that can be associated with a simple statement. To calculate the route and give an instruction, there is an action that the agents Theia (software) and speaker (hardware) must perform. As discussed earlier, each action that an agent performs, can be associated with one or more risks. Hence, the resulting risks could be Theia does not calculate the route and no instruction is given, or Theia calculates wrong route, etc. Alternatively, the route calculation by Theia may be perfect but the speaker may not give out the instruction. We identify templates using rules, implement these templates in our tool, to generate risks when a requirement, specification or domain statement is provided.

***Step 4*:** Risks obtained from step 3 are prioritized using the ontology. All the risks are compared against the set of risks listed in the domain specific, class-level ontology. The risks listed in the ontology are prioritized and the risks that are irrelevant (commissions) are ignored.

Step 5: *Risk-mitigation techniques* are devised based on the risks prioritized in step 4 which are shown in purple in fig 3. The break/hurt arrow represents that a risk mitigation technique hurts the risk, and it prevents that risk from happening. For e.g., if we prioritize the risk – the speaker may not give out the instruction since the speaker does not work, the risk mitigation technique that may help avoid that risk is to include a test voice/music clip which can be played by the user to make sure that the speaker is working before indicating the destination, etc.

D. Murphy Assistant tool: Murphy Assistant is a semi-automated Risk Analysis tool, where the user of the application has to setup the ontology before performing risk identification and analysis. For this process, we developed a windows application using the .NET framework. For storing all ontological concepts entered by the user, a Microsoft SQL Server Local Database is used. Murphy Assistant is a prototype tool which supports the concepts of Murphy framework. The refinement rules are provided as templates to this tool, and these templates are semantically bound. The underlying code can and the snapshots of the tool in action can be found at https://github.com/indoornavigation0/Murphy.git

IV. Experimentation

To validate our risk analysis and to identify the strengths and weaknesses of Murphy, we design an experiment to develop a smartphone app from the results obtained by performing risk analysis using Murphy.

***A. Experimental Setup*:** Murphy is intended to be used by requirements engineers and developers to perform risk analysis during the software development life cycle before the development of the application. We have conducted experiments, to validate Murphy with the help of a group of 25 PhD and 25 senior-level graduate students. All the students majored in computer science. Every student was provided with a version of Murphy installed on their computer. The students were given the requirement that we used as the running example, and tested many different requirements of their choice, chose the branches for which risk analysis should be performed, chose the rules to be applied to requirement, specification, or domain and when to stop the risk analysis (depth). After using Murphy, the students provided us with the list of risks and their feedback regarding the ease of use, accuracy of the automation and its usability, along with a list of risks identified.

A version of Theia has been developed using the list of risks and risk mitigation strategies provided by the students. One such mechanism has been implemented in Theia. We have conducted experimentation using Theia with 25 students.

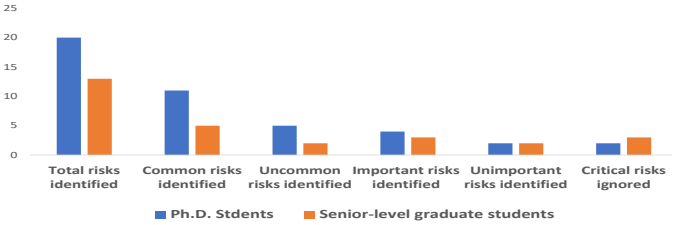

Figure 4: Different categories of risks identified using Murphy

***B. Analysis of the result*:** Performing risk analysis is a vital step before the development. Exclusion of RAG, as per our observation, we feel, shows that some risks are omitted. All the students were able to find common risks related to agents malfunctioning. We have observed various kinds of risks where the system did not behave the way it was supposed to. We also identified some risks where the system's functionality was aberrant. There were some risks which were not very relevant to the domain.

Students identified risks such as missing route, walking in the wrong direction, etc., critical risks such as falling, bumping into people, colliding against walls, etc., uncommon risks such as oil on the floor, banana peel on the way, water puddle on the floor, etc., unimportant risks such as warnings which ask the user to increase the volume, increase screen brightness, etc. The students have ignored some critical risks such as low battery indication, faulty voice input due to background noise, walking in a zig-zag fashion in a straight corridor, the blind person walking into a busy intersection, the user walking wrong number of steps, etc. These results are discussed in fig. 4, a bar graph which shows the number of risks in different categories identified by both the Ph.D. and Graduate-level students. Fig. 5 shows a bar graph, which compares the results of the risks faced/ignored while using the version of Theia developed without Murphy versus the version of Theia developed with Murphy.

In the version of Theia developed using results from Murphy, the system counts the step as the user walks, to help ensure safety of the user and to keep track of the steps walked by the user. This is the biggest difference between the app developed by using results from the Murphy versus not using the results from Murphy. Based on our observation, we feel that the students who used Theia (developed using Murphy's results) were able to walk very confidently since Theia was counting steps for them while navigating. Most of the students were able to make an accurate turn at the right spot, were able to keep track of their steps and were able to enjoy the process of navigation with ease. Overall, we feel that the results observed from these experiments show us that performing risk analysis during requirements engineering can help the end user avoid risks to some extent.

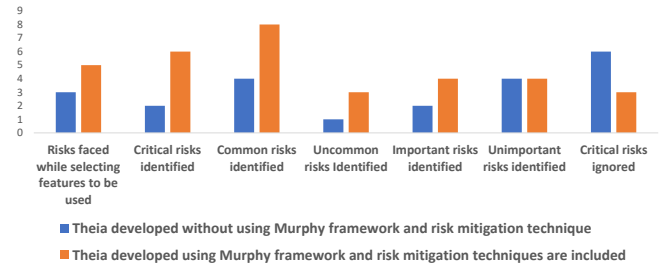

Figure 5: Results observed while using Theia developed without using Murphy vs using Murphy framework

C. Threats to Validity: We feel that our experiments have shown that there is a need to improve existing smartphone apps and devices for blind people, especially with features that ensure that the blind person is comfortable while navigating using the smartphone app, by building a tool for identifying risks systematically, devising anti-risk mechanisms and incorporating those results into the system before development. The Murphy Assistant tool needs improvements with more rules and templates to identify more complicated and uncommon risks. Since both Murphy Assistant and Theia are tested by students, and since the knowledge of the students is limited, the results may vary greatly compared to the app being tested by requirements engineers. The risks identified also varied greatly from what we anticipated since the use of the tool is based on individual knowledge and the way of performing analysis varies from person to person, therefore our results suffered. We are yet to test our Theia app with real blind people as we are yet to receive our IRB approval. We feel that testing with real blind people may give us an edge over blindfolded people, especially with identifying a variety of risks they face.

V. CONCLUSION

In this paper, we presented Murphy - a framework for performing risk identification and analysis using Augmented Reference Model - The Reference Model augmented with risks that was extended drawing to our previous work [19]. In this paper, we presented: 1. An activity-oriented ontology to perform risk analysis, 2. Risk Analysis Graph - for identifying and prioritizing and to devising risk mitigation techniques, 4. A tool, Murphy Assistant developed as a proof of concept, to generate RAGs for different requirements, specifications, and domains. 5. A reference application Theia is used to evaluate the strengths and weaknesses of Murphy. Based on the feedback from the students who used Murphy, we feel that its use during requirements engineering can indeed help increase the confidence of the engineers and developers in identifying some critical risks.

As future work, we plan to apply our approach to a wide variety of domains (e.g., autonomous vehicles domain) for performing risk analysis and providing risk mitigation strategies. We are developing a set of rules which aid in risk identification and analysis which goes beyond logic (simple negation). Experimentation of Theia with real blind subjects will be performed once we obtain the IRB approval. A step-by-step approach for engineers to develop and design their own graphically oriented Risk Analysis Graph's (RAGs) and identifying risks is underway as well. Finally, we plan to include safety and timeliness as a softgoal and extend our work using a goal-oriented approach.

REFERENCES

[1] Asnar, Y., Giorgini, P. Mylopoulos, J. Goal-driven risk assessment in requirement engineering. Requirements Eng 16, 101–116 (2011). https://doi.org/10.1007/s00766-010-0112-x

[2] Murphy's Law, https://en.wikipedia.org/wiki/Murphy's_law , Last accessed 2 February 2022

[3] Sharma, K., Kumar, P.V.. (2014). A method to risk analysis in requirement eng neering through optimized goal selection tropos goal layer. Journal of Theoretical and Applied Information Technology. 61. 270- 280.

[4] Cailliau, A. Lamsweerde, A. V.. "A probabilistic framework for goal-oriented risk analysis,"(2012). 20th IEEE International Requirements Engineering Conference (RE). Chicago, IL, 2012, pp. 201-210. doi: 10.1109/RE.2012.6345805.

[5] Lamsweerde, A. V.. "Risk-driven Engineering of Requirements for Dependable

[6] Gunter, C. A., Gunter, E. L., Jackson, M. Zave, P."A reference model Systems." En gineering Dependable Software Systems for requirements and specifications," in IEEE Software, vol. 17, no. 3, pp. 37-43, May-June 2000, doi: 10.1109/52.896248.

[7] Zave, P., Jackson, M.. (1997). Four dark corners of requirements engineering. ACM Trans. Softw. Eng. Methodol. 6, 1 (Jan. 1997), 1–30. DOI:https://doi.org/10.1145/237432.237434

[8] Vraalsen F., den Braber F., Lund M.S., Stølen K. (2005) The CORAS Tool for Security Risk Analysis. In: Herrmann P., Issarny V., Shiu S (eds) Trust Manage ment. iTrust 2005. Lecture Notes in Computer Science, vol 3477. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11429760_30

[9] Lamsweerde, A.V.. (2013). Risk-driven engineering of requirements for dependable systems. 10.3233/978-1-61499-207-3-207.

[10] Mylopoulos, J. Castro, J.."Tropos: A Framework for Requirements- Driven Software Development," (2000).INFORMATION SYSTEMS ENGINEERING: STATE OF THE ART AND RESEARCH THEMES, pp. 261-273.

[11] Cailliau, A., van Lamsweerde, A. Assessing requirements-related risks through probabilistic goals and obstacles. Requirements Eng 18, 129–146 (2013). https://doi.org/10.1007/s00766-013-0168-5

[12] Sales T.P., Baião F., Guizzardi G., Almeida J.P.A., Guarino N., Mylopoulos (2018) The Common Ontology of Value and Risk. In: Trujil.lo J. et al. (eds) Conceptual Modeling. ER 2018. Lecture Notes in Computer Science, vol 11157. Springer, Cham. https://doi.org/10.1007/978-030-00847-5_11

[13] Rosa, E.. "Metatheoretical foundations for post-normal risk." Journal of Risk Research 1 (1998): 15-44.

[14] Greenspan, S., Mylopoulos, J. Borgida, A.. 1994. On formal Requirements modeling languages: RML revisited. In Proceedings of the 16th international conference on Software engineering (ICSE '94). IEEE Com. puter SocietyPress, Washington, DC, USA, 135–147.

[15] Contrapositive, https://en.wikipedia.org/wiki/Contraposition". Last accessed 29 September 2021

[16] Ishikawa, K.: Introduction to quality control. Productivity Press (1990)

[17] Vesely, B.: Fault tree analysis (fta): Concepts and applications. NASA HQ(2002)

[18] Supakkul, S., Chung, L.: Extending problem frames to deal with stake Holder problems: An agent-and goal-oriented approach. In: Proceedings of the 2009 ACM symposium on Applied Computing. (2009) 389-394

[19] K. Kolluri, R. Ahn, T. Hill, L. Chung, " Risk Analysis for Collaborative Systems during Requirements Engineering", " Proc., International Conference on Software Engineering & Knowledge Engineering (SEKE 2021). 2021, pp. 297-302.]