Schematic Generation of English-prose Semantics for a Risk Analysis Language Based on UML Interactions

Gencer Erdogan*†, Atle Refsdal* and Ketil Stølen*†
*Department for Networked Systems and Services, SINTEF ICT,
PO Box 124 Blindern, N-0314 Oslo, Norway
Email: {gencer.erdogan, atle.refsdal, ketil.stolen}@sintef.no
†Department of Informatics, University of Oslo,
PO Box 1080 Blindern, N-0316 Oslo, Norway

***Abstract*—To support risk-driven testing, we have developed CORAL, a language for risk analysis based on UML interactions. In this paper, we present its semantics as a translation of CORAL diagrams into English prose. The CORAL semantics is developed to help software testers to clearly and consistently document, communicate and analyze risks in a risk-driven testing process. We first provide an abstract syntax and a translation algorithm. Then, we evaluate the approach based on some examples. We argue that the resulting English prose is comprehensible by testers, is consistent with the semantics of UML interactions, and has a complexity that is linear to the complexity of CORAL diagrams in terms of size.**

***Keywords*-risk analysis language; risk-driven testing; UML interaction; sequence diagram; CORAL diagram;**

I. Introduction

In earlier work, we presented a systematic method for designing test cases by making use of risk analysis [1], [2]. As part of the method, we also introduced a risk analysis language based on UML interactions which we refer to as CORAL. CORAL extends UML interactions with constructs for representing risk-related information in sequence diagrams, and it is specifically developed to support software testers in a risk-driven testing process.

As we explain in [1], [2], testers may use CORAL in three consecutive steps to identify, estimate, and evaluate risks. The graphical icons representing risk-related information in CORAL are based on corresponding graphical icons in CORAS [3]. This is a deliberate design decision because the graphical icons in CORAS are empirically shown to be cognitively effective [4]. However, without supporting natural-language semantics, CORAL diagrams, i.e., interactions represented by CORAL constructs, may be interpreted differently by different testers. Thus, in order to help software testers to clearly and consistently document, communicate and analyze risks, we present a structured approach to generate the semantics of CORAL diagrams in terms of English prose. We evaluate the approach based on some examples.

The remainder of this paper is organized as follows. Section II lists the success criteria our approach aims to fulfill. Section III gives a stepwise explanation of the approach, and presents the examples we base our evaluation on. Section IV elaborates on the fulfillment of the success criteria. Section V provides an overview of related work. Finally, Section VI gives some concluding remarks.

II. Success criteria

There are three key design decisions that shape our success criteria.

First, the main target audience of CORAL is software testers. CORAL is supposed to be used by testers to document, communicate and analyze risks in a risk-driven testing process. Thus, our first success criterion is: The resulting English prose must be comprehensible by software testers when conducting risk analysis.

Second, CORAL is based on UML interactions and only *extends* UML interactions with constructs representing risk-related information. Thus, our second success criterion is: The CORAL semantics of the constructs inherited from UML interactions must be consistent with their semantics in the UML standard.

Third, the approach must ensure scalability. Thus, our third success criterion is: The complexity of the resulting English prose must scale linearly with the complexity of CORAL diagrams in terms of size.

III. Approach

Inspired by CORAS [3], we generate the English-prose semantics in three consecutive steps, as shown in Figure 1. In **Step 1**, we translate a CORAL diagram into a corresponding textual representation. This step takes a CORAL diagram as input. First, for each construct in the CORAL diagram, we identify its corresponding syntactical element in the abstract syntax of CORAL. Second, we replace the variables in the syntactical element with content from the construct in the diagram. The output of this step is a textual representation of the CORAL diagram given as input to the step. The abstract syntax of CORAL is defined in Section III-A.

In **Step 2**, we translate the textual representation of a CORAL diagram into English prose, by making use of the

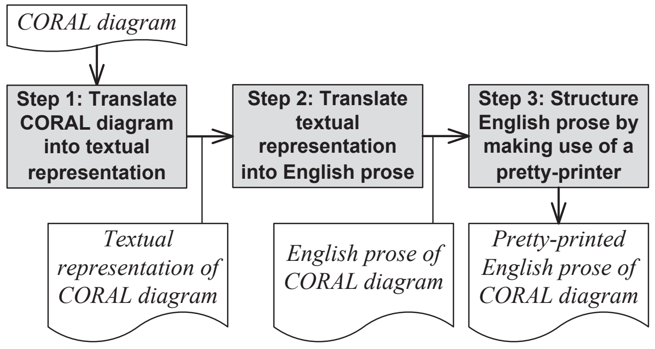
CORAL diagram
Step 1: Translate CORAL diagram into textual representation
Step 2: Translate textual representation into English prose
Step 3: Structure English prose by making use of a pretty-printer
Textual representation of CORAL diagram
English prose of CORAL diagram
Pretty-printed English prose of CORAL diagram

Figure 1. Generating English-prose semantics of CORAL diagrams.

translation algorithm defined in Section III-B. The translation algorithm is defined in terms of a function that takes syntactical elements as input and provides their translation as English prose.

Before presenting the translation function, we need to explain weak sequencing, which is a key construct in UML interactions. Weak sequencing is the implicit composition mechanism combining the constructs of an interaction, and is defined as follows [5]:

1) The transmission of a message must occur before its reception.
2) Events on the same lifeline are ordered in time, where time proceeds from the top of the lifeline towards the bottom of the lifeline, and where an event is either the transmission of a message or the reception of a message.

In the translation function, we use the term '*weakly sequenced by*' to denote weak sequencing as defined above. In **Step 3**, we make use of a pretty-printer to format the English prose in a structured manner. The technical details of such a pretty-printer are outside the scope of this paper, and are therefore not discussed here.

A. Abstract Syntax of CORAL

In this section, we define the abstract syntax of CORAL expressed in the Extended Backus-Naur Form [6]. The syntax defined in this section is an excerpt of the complete syntax, but it is sufficient for walking through the examples in the paper. The complete syntax is given in a technical report [7].

We use the following undefined terms in the grammar: $identifier$, $asset\ lifeline$, $frequency$, $conditional\ ratio$, and $consequence$. The term $identifier$ is assumed to represent any alphanumeric string. The term $asset\ lifeline$ is assumed to represent an alphanumeric string describing the name of an asset lifeline. The term $frequency$ is assumed to represent an alphanumeric string describing a likelihood value in terms of frequency. The term $conditional\ ratio$ is assumed to represent an expression that evaluates to a subset of $\mathbb{R}_{\geq 0}$ (the non-negative real numbers including 0). The term $consequence$ is assumed to represent an alphanumeric string describing the impact an unwanted incident has on an asset.

$$\begin{aligned} risk\ interaction = {} & message \mid weak\ sequencing \\ & \mid potential\ alternatives \\ & \mid referred\ interaction \\ & \mid parallel\ execution \\ & \mid frequency\ assignment \\ & \mid conditional\ ratio\ assignment \\ & \mid consequence\ assignment; \end{aligned}$$

$$\begin{aligned} message = {} & risky\ message \\ & \mid unwanted\ incident\ message; \end{aligned}$$

$$\begin{aligned} risky\ message = {} & \textbf{rm}(identifier,\ transmitter\ lifeline, \\ & receiver\ lifeline,\ message\ category); \end{aligned}$$

$$\begin{aligned} unwanted\ incident\ message = {} & \textbf{uim}(identifier, \\ & transmitter\ lifeline, \\ & asset\ lifeline); \end{aligned}$$

$$\begin{aligned} transmitter\ lifeline = {} & general\ lifeline \\ & \mid deliberate\ threat\ lifeline; \end{aligned}$$

$$\begin{aligned} receiver\ lifeline = {} & general\ lifeline \\ & \mid deliberate\ threat\ lifeline; \end{aligned}$$

$$general\ lifeline = \textbf{gl}(identifier);$$

$$deliberate\ threat\ lifeline = \textbf{dtl}(identifier);$$

$$message\ category = non\text{-}manipulative \mid manipulative;$$

$$non\text{-}manipulative = \text{general};$$

$$manipulative = \text{new} \mid \text{alter};$$

$$weak\ sequencing = \textbf{seq}(\{risk\ interaction\}^-);$$

$$potential\ alternatives = \textbf{alt}(\{risk\ interaction\}^-);$$

$$referred\ interaction = \textbf{ref}(identifier);$$

$$parallel\ execution = \textbf{par}(\{risk\ interaction\}^-);$$

$$\begin{aligned} & frequency\ assignment = \\ & \quad \textbf{fa}(kind,\ risky\ message,\ frequency) \\ & \quad \mid \textbf{fa}(transmission,\ unwanted\ incident\ message, \\ & \quad frequency); \end{aligned}$$

$$kind = transmission \mid reception;$$

$$transmission = \text{!};$$

$$reception = \text{?};$$

$$\begin{aligned} & conditional\ ratio\ assignment = \\ & \quad \textbf{cra}((kind,\ risky\ message),\ conditional\ ratio, \\ & \quad (kind,\ risky\ message)); \end{aligned}$$

$$\begin{aligned} & consequence\ assignment = \\ & \quad \textbf{ca}(unwanted\ incident\ message,\ consequence); \end{aligned}$$

B. English-prose Semantics of CORAL

The English-prose semantics of a syntactical element is defined by the function $[\![\ ]\!]$, which is defined below for the excerpt of the abstract syntax presented in Section III-A. Let the syntactical variables:

- d range over $risk\ interaction$
- id range over $identifier$
- t range over $transmitter\ lifeline$
- r range over $receiver\ lifeline$
- al range over $asset\ lifeline$
- f range over $frequency$
- cr range over $conditional\ ratio$
- c range over $consequence$

The pair of square brackets, '[' and ']', is a part of the semantics that is used to enclose an operand.

$[\![\textsf{seq}(d_1, d_2, .., d_m)]\!] =$ **[** $[\![d_1]\!]$ **]** weakly sequenced by **[** $[\![d_2]\!]$ **]** weakly sequenced by ... weakly sequenced by **[** $[\![d_m]\!]$ **]**

$[\![\textsf{alt}(d_1, d_2, .., d_m)]\!] =$ either **[** $[\![d_1]\!]$ **]** or **[** $[\![d_2]\!]$ **]** or ... or **[** $[\![d_m]\!]$ **]**

$[\![\textsf{ref}(id)]\!] =$ refer to interaction: id

$[\![\textsf{par}(d_1, d_2, .., d_m)]\!] =$ **[** $[\![d_1]\!]$ **]** parallelly merged with **[** $[\![d_2]\!]$ **]** parallelly merged with ... parallelly merged with **[** $[\![d_m]\!]$ **]**

$[\![\textsf{rm}(id, t, r, \text{general})]\!] =$ the message id is transmitted from $[\![t]\!]$ to $[\![r]\!]$

$[\![\textsf{gl}(id)]\!] = id$

$[\![\textsf{dtl}(id)]\!] =$ the deliberate threat id

$[\![\textsf{rm}(id, t, r, \text{new})]\!] =$ the new message id is transmitted from $[\![t]\!]$ to $[\![r]\!]$

$[\![\textsf{rm}(id, t, r, \text{alter})]\!] =$ the altered message id is transmitted from $[\![t]\!]$ to $[\![r]\!]$

$[\![\textsf{uim}(id, t, al)]\!] =$ the unwanted incident id occurring on $[\![t]\!]$ impacts asset al

$[\![\textsf{fa}(?, (id, t, r, \text{new}), f)]\!] =$
the reception of the new message id by $[\![r]\!]$ from $[\![t]\!]$ occurs with frequency f

$[\![\textsf{fa}(!, (id, t, al), f)]\!] =$
the unwanted incident id occurring on $[\![t]\!]$ impacts asset al with frequency f

$[\![\textsf{cra}((!, (id, t, r, \text{alter})), cr, (?, (id, t, r, \text{alter})))]\!] =$
the transmission of the altered message id from $[\![t]\!]$ leads to its reception by $[\![r]\!]$ with conditional ratio cr

$[\![\textsf{ca}((id, t, al), c)]\!] =$
the unwanted incident id occurring on $[\![t]\!]$ impacts asset al with consequence c

We demonstrate the schematic translation of CORAL diagrams into English prose by first giving some examples of CORAL diagrams (see Figure 2), and then translating these diagrams into their corresponding English prose using the translation functions (see Figure 3). The CORAL diagrams in Figure 2 were obtained by applying our method [1], [2] on a guest book that is available in the Damn Vulnerable Web Application [8].

IV. Discussion

In this section, we discuss the fulfillment of the three success criteria given in Section II.

A. The resulting English prose must be comprehensible by software testers when conducting risk analysis

The comprehensibility of the resulting English prose is supported both from a general viewpoint and from a software testing viewpoint.

From a general viewpoint, we observe the following two points. **First**, the structure of the translations in Figure 3 is similar to the structure of their corresponding CORAL diagrams in Figure 2. In particular, the ordering of the translated CORAL constructs is maintained. For example, let us consider the translation in Figure 3a. The first sentence states: "The new message forgedURLReplacingMsgWithXSSscript is transmitted from the deliberate threat Hacker to C". By comparing the translation in Figure 3a to its corresponding diagram in Figure 2a, we see that the first sentence corresponds to the first message in the diagram. Similarly, we see that the second sentence in Figure 3a corresponds to the second message in Figure 2a, and so on. **Second**, the user-defined text is unchanged in the translations. By user-defined text, we mean the text typed in CORAL diagrams, such as the text on messages, lifelines, frequency assignments, consequence assignments, and so on.

From a software testing viewpoint, we observe that risk-related concepts from CORAL are integrated with concepts from UML interactions, in the resulting English prose. UML interaction is among the top three modeling language within the testing community and is often used for testing purposes [9]. It is therefore reasonable to assume that testers understand the concepts from UML interactions. Moreover, we find it reasonable to assume that testers also comprehend the risk-related concepts we introduce in CORAL, such as *altered* messages and messages representing *unwanted incidents*, because these are concepts that are also known within the testing community. For example, in fuzz testing, the expected behavior of a system is altered by providing invalid, unexpected, or random data, which may lead to unwanted incidents [10].

To illustrate this, let us consider the first message in Figure 2d. This message represents an altered message. In CORAL, an altered message is a message in the system model which has been altered due to unexpected system

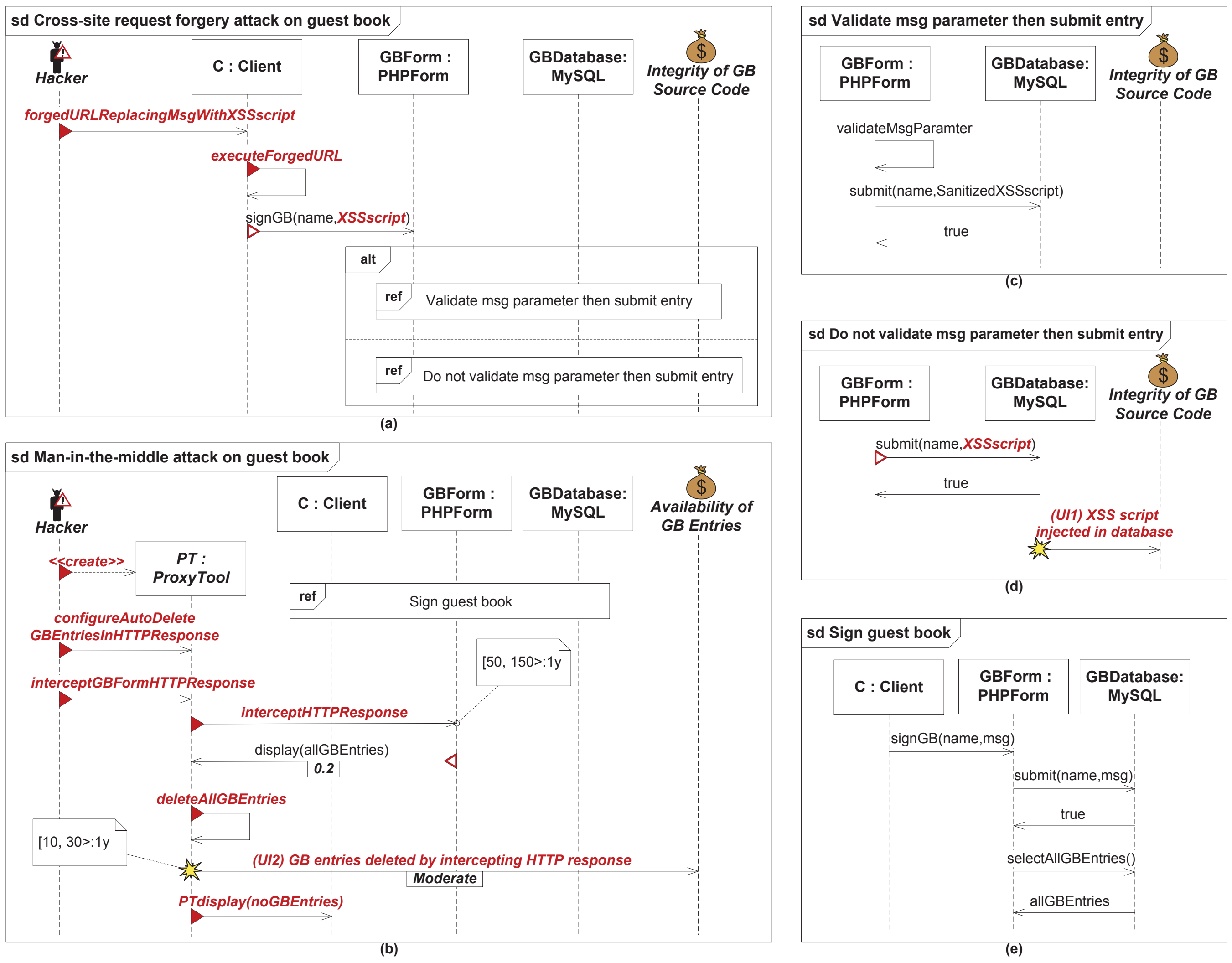
sd Cross-site request forgery attack on guest book
Hacker
C : Client
GBForm : PHPForm
GBDatabase: MySQL
Integrity of GB Source Code
forgedURLReplacingMsgWithXSSscript
executeForgedURL
signGB(name,XSSscript)
alt
ref
Validate msg parameter then submit entry
ref
Do not validate msg parameter then submit entry
(a)
sd Man-in-the-middle attack on guest book
Hacker
C : Client
GBForm : PHPForm
GBDatabase: MySQL
Availability of GB Entries
<<create>>
PT : ProxyTool
ref
Sign guest book
configureAutoDelete GBEntriesInHTTPResponse
[50, 150>:1y
interceptGBFormHTTPResponse
interceptHTTPResponse
display(allGBEntries)
0.2
deleteAllGBEntries
[10, 30>:1y
(UI2) GB entries deleted by intercepting HTTP response
Moderate
PTdisplay(noGBEntries)
(b)
sd Validate msg parameter then submit entry
GBForm : PHPForm
GBDatabase: MySQL
Integrity of GB Source Code
validateMsgParamter
submit(name,SanitizedXSSscript)
true
(c)
sd Do not validate msg parameter then submit entry
GBForm : PHPForm
GBDatabase: MySQL
Integrity of GB Source Code
submit(name,XSSscript)
true
(UI1) XSS script injected in database
(d)
sd Sign guest book
C : Client
GBForm : PHPForm
GBDatabase: MySQL
signGB(name,msg)
submit(name,msg)
true
selectAllGBEntries()
allGBEntries
(e)

Figure 2. Examples of CORAL diagrams.

behavior or unexpected input data. Figure 3d shows the corresponding translation as: "The altered message submit(name,XSSscript) is transmitted from GBForm to GBDatabase". The translation shows that we have a *message* that is transmitted between two *lifelines* (concepts from UML interactions), and we have added that the message is *altered* (risk-related concept from CORAL).

B. The CORAL semantics of the constructs inherited from UML interactions must be consistent with their semantics in the UML standard

The CORAL constructs inherited from UML interactions are messages and the interaction operators: seq, ref, alt, par and loop. The interaction operator **weak sequencing (seq)** is defined and related to CORAL in Section III.

According to UML [5], a **message** defines a particular communication between lifelines of an interaction. The signature of a message is the content specification of that very message. A message also defines the lifeline from which it is sent (i.e., the transmitter lifeline) and the lifeline receiving it (i.e., the receiver lifeline). Thus, a message may be defined as the triple (id, t, r), where id represents the signature, t represents the transmitter lifeline, and r represents the receiver lifeline. We define a message in a similar manner. However, we also distinguish between the category of a message, i.e., whether it is manipulative, non-manipulative, or an unwanted incident, as explained in Section III. As we can see from the translations in Figure 3, the English prose of messages are generated according to their category, and contain information about the message signature, the lifeline transmitting the message and the lifeline receiving the message.

According to UML [5], an **interaction-use (ref)** refers to an interaction. The interaction-use is shorthand for copying the contents of the referred interaction where the interaction-

Schematic translation of CORAL diagram: Cross-site request forgery attack on guest book.

[
The new message forgedURLReplacingMsgWithXSSscript is transmitted from the deliberate threat Hacker to C.
]
Weakly sequenced by [
The new message executeForgedURL is transmitted from C to C.
]
Weakly sequenced by [
The altered message signGB(name,XSSscript) is transmitted from C to GBForm.
]
Weakly sequenced by [
Either [
Refer to interaction: Validate msg parameter then submit entry.
]
or [
Refer to interaction: Do not validate msg parameter then submit entry.
]
]

(a)

Schematic translation of CORAL diagram: Man-in-the-middle attack on guest book.

[
Refer to interaction: Sign guest book.
]
Weakly sequenced by [
The new message «create» is transmitted from the deliberate threat Hacker to PT.
]
Weakly sequenced by [
The new message configureAutoDeleteGBEntriesInHTTPResponse is transmitted from the deliberate threat Hacker to PT.
]
Weakly sequenced by [
The new message interceptGBFormHTTPResponse is transmitted from the deliberate threat Hacker to PT.
]
Weakly sequenced by [
The new message interceptHTTPResponce is transmitted from PT to GBForm. The reception of the new message interceptHTTPResponce by GBForm from PT occurs with frequency [50, 150>:1y.
]
Weakly sequenced by [
The altered message display(allGBEntries) is transmitted from GBForm to PT. The transmission of the altered message display(allGBEntries) from GBForm leads to its reception by PT with conditional ratio 0.2.
]
Weakly sequenced by [
The new message deleteAllGBEntries is transmitted from PT to PT.
]
Weakly sequenced by [
The unwanted incident (UI2) GB entries deleted by intercepting HTTP response occurring on PT impacts asset Availability of GB Entries with frequency [10, 30>:1y. The unwanted incident (UI2) GB entries deleted by intercepting HTTP response occurring on PT impacts asset Availability of GB Entries with consequence Moderate.
]
Weakly sequenced by [
The new message PTdisplay(noGBEntries) is transmitted from PT to C.
]

(b)

Schematic translation of CORAL diagram: Validate msg parameter then submit entry.

[
The message validateMsgParameter is transmitted from GBForm to GBForm.
]
Weakly sequenced by [
The message submit(name,SanitizedXSSscript) is transmitted from GBForm to GBDatabase.
]
Weakly sequenced by [
The message true is transmitted from GBDatabase to GBForm.
]

(c)

Schematic translation of CORAL diagram: Do not validate msg parameter then submit entry

[
The altered message submit(name,XSSscript) is transmitted from GBForm to GBDatabase.
]
Weakly sequenced by [
The message true is transmitted from GBDatabase to GBForm.
]
Weakly sequenced by [
The unwanted incident (UI1) XSS script injected in database occurring on GBDatabase impacts asset Integrity of GB Source Code.
]

(d)

Schematic translation of CORAL diagram: Sign guest book

[
The message signGB(name,msg) is transmitted from C to GBForm.
]
Weakly sequenced by [
The message submit(name,msg) is transmitted from GBForm to GBDatabase.
]
Weakly sequenced by [
The message true is transmitted from GBDatabase to GBForm.
]
Weakly sequenced by [
The message selectAllGBEntries() is transmitted from GBForm to GBDatabase.
]
Weakly sequenced by [
The message allGBEntries is transmitted from GBDatabase to GBForm.
]

(e)

Figure 3. Schematic translation of the corresponding CORAL diagrams in Figure 2 into English prose.

use is. Figure 2b shows an example of an interaction-use named Sign guest book. The interaction referred to by this interaction-use is shown in Figure 2e. We use the term '*refer to interaction*' to denote an interaction-use, as shown in the translations in Figures 3a and 3b.

According to UML [5], the interaction operator **potential alternatives (alt)** designates that the operands represent a choice of behavior. UML requires that the chosen operand must have an explicit or implicit guard expression that evaluates to true. An implicit true guard is implied if the operand has no explicit guard. In CORAL, we currently allow only for the usage of implicit true guards. However, the syntax and semantics of CORAL is easily extendable to support explicit guards as well. As shown in Figure 3a, we use the term '*either*' in front of the first operand of an alt operator, and then the term '*or*' between each subsequent operand to reflect the disjunctive behavior of the alt operator.

According to UML [5], the interaction operator **parallel execution (par)** designates a parallel merge between the behaviors of the operands. A parallel merge defines a set of traces that describes all the ways that events of the operands may be interleaved without obstructing the order of the events within the operands. We use the term '*parallelly merged with*' between each operand to denote a parallel merge between the behaviors of the operands.

C. The complexity of the resulting English prose must scale linearly with the complexity of CORAL diagrams in terms of size

As illustrated by Figure 2 and Figure 3, the definition of the translation function in Section III-B ensures that the structure of its output mirrors the input diagram, and that there is a linear relationship between the size of input and output. A formal argument that this would hold for any diagram d could be given based on induction over the syntactical structure of d.

V. Related Work

To the best of our knowledge, no risk-driven testing approach provides a similar schematic generation of natural language semantics as presented in this paper. Most approaches use risk tables/matrices or risk annotated models as a means for documenting, communicating and analyzing risks. However, some approaches provide guidelines for documenting risk-related information in natural-language semantics. Redmill [11] provides a set of guide words with associated definitions, which may be used as a basis for documenting risk-related information. Gleirscher [12] provides a similar approach and makes use of a safety analysis pattern for describing informal test cases. Nazier and Bauer [13] provide a template for documenting safety-risk information, while Kumar et al. [14] provide a template for documenting risk-related information within the domain of aspect oriented programming. Souza et al. [15] use a taxonomy based questionnaire for documenting risk-related information.

VI. Conclusion

CORAL is a risk analysis language based on UML interactions, and it is specifically developed to support software testers in a risk-driven testing process. CORAL extends UML interactions with constructs for representing risk-related information in sequence diagrams.

In this paper, we presented a structured approach to generate the semantics of CORAL diagrams in terms of English prose. The CORAL semantics is developed to help testers to clearly and consistently document, communicate and analyze risks in a risk-driven testing process. In particular, it helps testers to: (1) obtain a correct understanding of CORAL diagrams, (2) analyze risks posed on the system under test in a clear and consistent manner, and (3) clearly communicate risks posed on the system under test.

We argue that the resulting English prose is comprehensible by testers because: (1) it preserves the structure of CORAL diagrams, (2) it keeps the user-defined text in CORAL diagrams unchanged, and (3) it uses concepts that are known to software testers. In addition, the resulting English prose of the constructs inherited from UML interactions is consistent with their semantics in the UML standard [5]. Moreover, the complexity of the resulting English prose scales linearly with the complexity of the CORAL diagrams in terms of size.

Acknowledgment

This work has been conducted as a part of the DIAMONDS project (201579/S10) funded by the Research Council of Norway, the NESSoS network of excellence (256980) and the RASEN project (316853) funded by the European Commission within the 7th Framework Programme, as well as the CONCERTO project funded by the ARTEMIS Joint Undertaking (333053) and the Research Council of Norway (232059).

References

[1] G. Erdogan, A. Refsdal, and K. Stølen, "A Systematic Method for Risk-Driven Test Case Design Using Annotated Sequence Diagrams," in *Proc. 1st International Workshop on Risk Assessment and Risk-driven Testing (RISK'13)*. Springer, 2014, pp. 93–108.

[2] ——, "A Systematic Method for Risk-Driven Test Case Design Using Annotated Sequence Diagrams," SINTEF Information and Communication Technology, Technical Report A26036, 2014.

[3] M. S. Lund, B. Solhaug, and K. Stølen, *Model-Driven Risk Analysis: The CORAS Approach*. Springer, 2011.

[4] B. Solhaug and K. Stølen, "The CORAS Language - Why it is designed the way it is," in *Proc. 11th International Conference on Structural Safety and Reliability (ICOSSAR'13)*. CRC Press, 2013, pp. 3155–3162.

[5] *Unified Modeling Language (UML), superstructure, version 2.4.1*, Object Management Group, 2011, OMG Document Number: formal/2011-08-06.

[6] *ISO/IEC 14977:1996(E), Information technology – Syntactic metalanguage – Extended BNF, first edition*, International Organization for Standardization, 1996.

[7] G. Erdogan, A. Refsdal, and K. Stølen, "Schematic Generation of English-prose Semantics for a Risk Analysis Language Based on UML Interactions," SINTEF Information and Communication Technology, Technical Report (to appear), 2014.

[8] "Damn Vulnerable Web Application," accessed September 16, 2014. [Online]. Available: http://www.dvwa.co.uk/

[9] A. D. Neto, R. Subramanyan, M. Vieira, and G. Travassos, "A Survey on Model-based Testing Approaches: A Systematic Review," in *Proc. 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies (WEASELTech'07)*. ACM, 2007, pp. 31–36.

[10] P. Oehlert, "Violating assumptions with fuzzing," *Security Privacy, IEEE*, vol. 3, no. 2, pp. 58–62, 2005.

[11] F. Redmill, "Theory and practice of risk-based testing," *Software Testing, Verification and Reliability*, vol. 15, no. 1, pp. 3–20, 2005.

[12] M. Gleirscher, "Hazard-based selection of test cases," in *Proc. 6th International Workshop on Automation of Software Test (AST'11)*. ACM, 2011, pp. 64–70.

[13] R. Nazier and T. Bauer, "Automated risk-based testing by integrating safety analysis information into system behavior models," in *Proc. 23rd International Symposium on Software Reliability Engineering Workshops (ISSREW'12)*. IEEE, 2012, pp. 213–218.

[14] N. Kumar, D. Sosale, S. N. Konuganti, and A. Rathi, "Enabling the adoption of aspects-testing aspects: A risk model, fault model and patterns," in *Proc. 8th ACM International Conference on Aspect-Oriented Software Development (AOSD'09)*. ACM, 2009, pp. 197–206.

[15] E. Souza, C. Gusmão, and J. Venâncio, "Risk-based testing: A case study," in *Proc. 7th International Conference on Information Technology: New Generations (ITNG'10)*. IEEE, 2010, pp. 1032–1037.