Simulation-based Completeness Analysis and Adaption of Fault Trees

Volker Gollücke
and Axel Hahn
System Analysis and Optimization,
Carl-von-Ossietzky-Universität Oldenburg,
Oldenburg, Germany
e-mail: {golluecke,hahn}@wi-ol.de

Jan Pinkowski, Christoph Läsche,
and Sebastian Gerwinn
OFFIS
Institute for Information Technology
Oldenburg, Germany
e-mail: {pinkowski,laesche,gerwinn}@offis.de

Abstract—**Safety analysis is a common and important task for any operational planning of missions such as, in a maritime context, the construction and maintenance of offshore wind farms. Identifying potential risks that might occur during the planned operation or a sequence of operations is the main task of an associated hazard and risk analysis. This paper introduces an approach to automatically adapt fault trees based on a simulation of a model of the operation in question enhanced by a formal description of hazards. Formalizing the corresponding hazard specifications allows us to generate observers which in turn identify failures and hazards during a simulation of the system. Such detection of critical situation can interact with the simulation via a specialized controller application for the simulation thereby triggering the completion of an existing fault tree. As potentially stochastic models of the environment are considered as well, this simulation approach naturally provides guarantees of the result in terms of statistical confidence statements. The feasibility of this approach is exemplified on an offshore lifting operation.**

***Keywords*–Simulation; Statistical Model Checking; Fault Trees; Risk Assessment; Offshore Operations; Observer.**

I. Introduction

Identifying and mitigating potential hazards is an important yet demanding task in nearly every system design process. The likelihood of missing a potential risk depends critically on the degree of complexity of the operation to be planned. Therefore, having means of checking the completeness of an initial risk picture is highly desirable. Current safety regulations, e.g., within the domain of offshore operations, require a description of all involved risks [1][2]. Specifically, these regulations require an in-depth characterization of risks, including listing causal factors as well as options for mitigation of high-level undesired events [3]. A common way to visualize and construct such characterizations of the involved risks are fault trees [4]. Fault trees split the risk of violating an overall safety goal, e.g., physical inviolability of all personnel, into different failures which in its combination might lead to the undesired event. Although fault trees are easy to interpret and can be used to calculate the probabilistic rate of the undesired event, they are usually created manually and therefore prone to oversight. On the other hand, simulations are widely used to optimize complex operations, as they often can give a more accurate estimate of incurred costs of an operation [5]. Supplementary to the manual process of creating risk descriptions in terms of fault trees, we propose a simulation-based risk analysis which can be used to complete a given fault tree by iteratively adding failure-combinations which have been detected by the simulation to be able to trigger a violation of the overall safety goal. Importantly, this approach comes with a statistical statement which specifies the level of confidence that no further critical failure combinations exist. In particular, this includes a confidence statement about the probability of the occurrence of each top-level hazard being below a critical value. On the implementation side, we develop an integrated framework of high-level process planning, a platform for handling different co-simulations for different components of operations, and a tool to formalize the manually identified hazards and failure combinations to be able to detect them during simulation. We demonstrate the potential benefits of such a framework on a hazard and risk analysis problem for a specific offshore crane operation of moving a cargo on a vessel observed by a lift supervisor.

The paper is organized as follows. In Section II, we explain each component of the developed simulation framework including the used models, the generation of observers as well as the configuration of the simulation to be suitable for a statistical evaluation of the obtained simulation runs. To illustrate the framework, we present the evaluation of the framework on the considered use case in Section III before concluding in Section IV.

II. Methodology

Using simulation to identify additional failure combinations that lead to a hazard requires an accurate description of the operation to be performed and analyzed. This also means that potential risks have to be encoded into such model used for simulation – at least implicitly. Planning operations on a detailed level consists of planning many sub-tasks. As a result, undesired high level situations, such as personnel injury, might depend on complex circumstances which are not easily identified. We illustrate the proposed process for modeling, definition of risks, and integration with the simulation in Figure 1 which we will first describe on a high level before explaining the individual steps in more detail.

As a first step, the model for the operation to be performed has to be synthesized, as indicated in Figure 1 by *“System Model”*. Each of the defined sub-tasks can be annotated with potential failure combinations leading to hazards from which a *“Fault Tree Model”* can be synthesized. In parallel, the planned operation (and sub-tasks) can be refined iteratively, resulting in a *“Scenario Description Model”* which can be simulated. By using a suitable formalization of individual failures within the fault tree, observers can be generated. They are able to observe a simulation, indicating if an individual failure, hazard, or a combination thereof has occurred in a single simulation run. To this end, the *“Observer”* has to be triggered by a central *“Simulation and Control”* part. If an observer indicates failures that are currently not considered within the fault tree, a combination of them can be used to adapt the current fault tree, resulting in a more complete risk characterization. Otherwise, we can generate more simulation runs, until sufficiently many traces of the operation to be planned have been simulated without observing any non-considered failure combination.

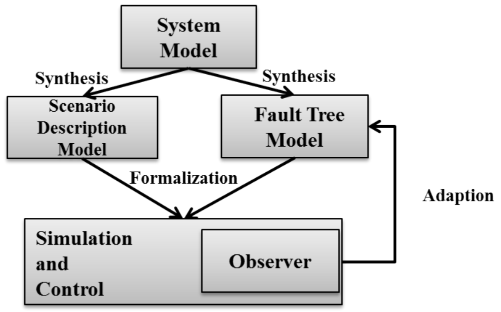


Figure 1. Schematic overview of the simulation-based analysis and adaption approach.

A. System Model

As a basis for our approach, we require a model of the system to be analyzed. We use a process model (cf. Droste et al. [6]) based on concepts from business process modeling languages such as BPMN [7] or the activity diagram of UML [8]. The respective process represent actors, tasks, and corresponding interactions. This model type has been developed for application in the maritime domain, more specifically for use for offshore operations. An exemplary model is described in Section III. The graphical elements of the model depict the task sequences and interactions and it is also possible to annotate corresponding hazards and failures to the model. A hazard describes a potential source of risks on a system-wide level, i.e., the injury or death of a person, the pollution of the environment, or the damage of systems. A failure describes a possible cause for a hazard or another, higher-level failure. In the proposed methodology, we use a formal specification for hazards and failures as described by Läsche et al. [9]. Additionally to the process model, the system model consists of an environment model which describes the considered resources like actors, physical objects, or environmental conditions. These resources are also accessible by the process model to map actors to specific avatars of the simulated environment. For example, a crane operator is mapped to a crane operator resource from the simulated environment with an orientation, a location, a mass, and further necessary attributes.

B. Scenario Description Model

In our case, a scenario description can be used to describe the course of a simulation. In our current approach it consists of

- Parameter bounds,
- Parameter exploration techniques,
- Exit conditions, and
- Different trigger types (i.e., simulation-time and simulation-step) to cause parameter changes or to define the end of a simulation (run).

Using the presented system model which describes the progress of an operation and the considered resources – like the ship, a crane, and a lift supervisor – and their properties, a scenario description template with default values can be automatically generated. The generation process reads the resources and their properties from the environment model as well as the modeled events and annotated failures and hazards from the process model. Based on this information the scenario description model is build. The build process categorizes the resources from the environment model and automatically sets a default trigger for each property. The default trigger is used to set the value of a property to the value defined in the environment model at the start of a simulation run. The scenario description is completed by event triggers which are extracted from the process model and are available to the modeler of a scenario.

After this automatic generation of the scenario description it can be adjusted. The adjustment allows to configure the parameter bounds of the resource properties like a minimum and maximum wave height, the parameter exploration techniques like a linear or random exploration and the configuration of the parameter change trigger and exit conditions which are used to define the end of a simulation or simulation run. The triggers can be based e.g., on the simulation time but also on the occurrence of situations such as the start of a task. These situations are defined in the process model. The completed scenario description can then be forwarded to the simulation control to configure the simulation runs (cf. Section II-E2).

C. Fault Tree Model

As part of risk assessment for offshore applications, Fault Tree Analysis (FTA) gains more and more attention in the maritime domain (cf. Vinnem [10], Lavasani et al. [11], and Sutton [12]). Thus, FTA has been recommended as methodology in several guidelines for risk assessment. However, fault trees are typically constructed manually. It requires experienced engineers to structure the system knowledge and necessary details. Manual construction of fault trees thus is time-consuming, cost-intensive, incomplete, and error-prone (cf. McKelvin et al. [13] and Tajarrod et al. [14]). Therefore, several approaches to automatically synthesize fault trees are published (cf. Chen [15] and Pai et al. [16]). These approaches, focusing on the construction of fault trees, typically use formal system models. If both, the underlying model and the individual fault tree elements, such as failures and hazards, are formalized, a formal completeness analysis can be performed via model checking (cf. Schellhorn et al. [17] and Ortmeier et al. [18] for more details on such a formal completeness analysis). By verifying these completeness conditions over the formalized fault tree via model checking, it is possible to guarantee that no additional failure as a cause for the specified system hazard has been missed within a given fault tree. However, this kind of completeness analysis typically does not scale well with the system's complexity. Furthermore, some parts of the system might only be given as a black box model for which no formal description is available but which can be simulated to generate traces describing its behavior. Thus, we focus on a simulation-based methodology to check fault trees completeness and provide a confidence statement about the statistical certainty of the existence of unconsidered failures. Thereby, the fault trees are automatically synthesized from the process model part of the system model. Process model elements such as tasks or actors have annotated hazards and failures. For each annotated hazard, we create a fault tree to logically structure the respective failures that could possibly cause the hazard. Since each hazard relates to an element of the process model, i.e., a task, the respective fault tree is also related to this task. Conversely, each top event of a fault tree corresponds to one hazard. For example, a task can have several fault trees that graphically represent its hazards and corresponding failures. In contrast to other fault tree construction approaches mentioned above all necessary information are included in the process model. The process model elements, such as a task or an actor have specific hazards and failures that are modeled by an experienced safety expert.

D. Observer

To be able to identify the occurrence of the hazards and failures depicted in the fault tree during run-time, we need a formalized description of them. Their identification is important as we want to check the fault trees for errors and completeness. Specifically, we can use the identified events to evaluate each simulation step regarding whether a top event has occurred. If this is the case, it is checked against the fault tree if there is a mismatch between failures identified by the observer and the failures in the fault tree. If a failure in the fault tree is not detected, the failure might not be required for the event to happen. Thus, the fault tree structure might be faulty and has to be reassessed. On the other hand it is checked if failures occur that are not covered in the current fault tree structure identified for the top event. If this is the case, we have to add the detected failure to the fault tree of the top event in order to complete the fault tree structure. As this process is repeated, a more and more accurate fault tree is created and failures required for the hazard to occur are highlighted whereas optional failures are identified and can be corrected. This allows us to have a most accurate estimation (to a certain confidence level) of the involved risks (to the limits of the simulation).

In order to identify hazards and failures during the simulation, we attach observers, i.e., programs to detect the occurrence of events, to the simulation environment. To be able to create such observers, a formal description of the events has to exist. This is achieved through the Hazard Description Language (HDL) which has been developed by us for this exact purpose. The language allows the analyzing person to formalize hazard and failures in near-natural language, allowing persons without knowledge about formalization languages to describe the events. It is used to describe the events that display a hazard or failure to be able to detect them. There is a set of potential formalization patterns, allowing the person to describe several aspects of an event to make it unambiguous. However, we do not directly use the HDL formalization and thus transform it into the Object Constraint Language (OCL) [19]. Thus, the HDL can be seen as an interface for describing hazards and failures using OCL without deeper knowledge of the constraint language by defining a set of HDL patterns that are automatically transferable to OCL expressions. OCL is a widely used language; thus, its usage minimizes our effort as we can rely on a well-evaluated language and multiple implementations.

Every OCL expression is evaluated by a separate observer client that is constantly updated with data from the simulation by the controller (cf. Section II-E1). At each simulation step, all observers are triggered by the controller to automatically evaluate if an event has occurred. If this is the case, the event then is logged for further analysis. The simulation result is analyzed and incorporated as described above. In contrast to the automatic observation of the simulation, this is a manual process.

E. Simulation-based Analysis

Our goal is to determine the completeness of a fault tree with possible failure combinations leading to an overall hazardous event. To this end, we need to ensure that the cut-sets of the fault tree is minimal or that there are no other failure combinations which are able to trigger an overall hazardous event. As we are following a simulation-based approach, such a guarantee can only be of statistical nature. Therefore, before detailing the simulation setup, we present the theoretical foundation of determining the necessary number of simulation runs needed for gathering sufficient evidence of the completeness of the current risk description of the corresponding fault tree.

1) Statistical Evaluation of Simulations: Based on multiple runs of the simulation, it is of critical importance to quantify the confidence in the found failure and hazard combinations. In particular, when no more failure combinations leading to a hazardous situation are found using simulation runs, we are aiming at bounding the probability that there are indeed no more critical failure combinations possible. As we allow the underlying model to exhibit stochastic behavior, we cannot be absolutely certain that there are no more failure combinations based on finitely many simulations. However, we can bound the probability that a hazard is caused by a previously not considered failure or combination. To this end, we are using a well-known confidence statement. More precisely, if we assume that we will observe a failure combination with probability p, the probability of not observing a critical situation within N samples is given by

$$P(\text{no failures observed}) = (1-p)^N \tag{1}$$

This equation can be used to construct a confidence statement of the following form. Once we have observed N without any observable critical situation, the following holds true (see Annex D in part 7 of [20]):

Up to a confidence $1-\alpha$, the probability of the occurrence of a critical situation can be bound by $1-\sqrt[N]{\alpha}$.

Vice versa, we can also bound the necessary number of simulations needed to be sure – up to a confidence level of $1-\alpha$ – that there is no critical situation having an occurrence probability larger than p^+ by :

$$N \geq -\frac{\log(\alpha)}{p^+} \tag{2}$$

By using this confidence statement, we are compliant with the IEC61508 [20]. Note that p^+ is used here to bound the compound probability of the failure combination. That is, using N simulations we can statistically show, that there are no failure combinations which have a probability of occurring together larger than p^+. Taken together we have the following procedure to perform a statistical completeness check:

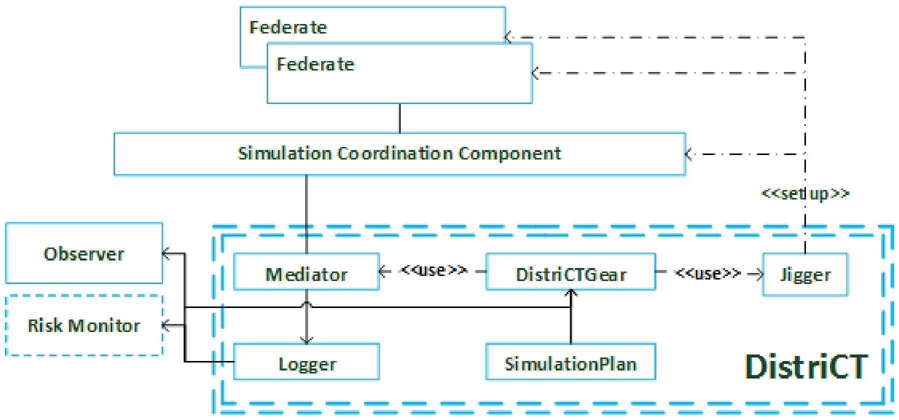


Figure 2. The structure of our distributed simulation setup. It consists of a simulation coordination component, federates (e.g., simulators), observer, and DistriCT to control and monitor the mentioned components.

1) Fix desired confidence level α and critical threshold for the probability p, for example $\delta = 0.05$ and $p = 10^{-8}$
2) Calculate number of simulation runs N
3) Generate N number of simulations
4) If all simulations do not reveal a previously not considered failure combination, we have reached the desired level of certainty of no additional failure combinations
5) If additional risks have been identified, return to 1.

Note, however, that the necessary number of simulations does not scale well with low thresholds for critical probabilities p (cf. equation (2)). In particular, the necessary number of simulation runs scales linearly with the critical threshold. For example, if we chose the critical probability threshold to be $p^+ < 10^{-9}$, roughly 10^9 simulation runs are needed. To be able to obtain feasible confidence statement in these kind of situations, specialized, guided simulation methods such as importance splitting or sampling [21][22][23] are needed. However, for the sake of simplicity, we use the procedure presented above. Having a formalization of the hazardous situations could also help to define a score function thereby defining a notion of proximity to a critical situation. This score function, for example could then in turn be used to guide the simulation toward the rare event (see [24]). These kind of simulations also require that the individual simulations can be controlled on an individual time step basis, which we will explain in the next section.

2) Simulation and Simulation Control: To be able to find overlooked failures based on simulation, we use a simulation setup and simulation control tool. Therefore the scenario description can be used together with a description of the required simulators to generate a simulation plan. The simulation plan itself describes a configured sequence of simulation runs and also accesses the simulation environment depicted in Figure 2. The proposed structure, also used in our use case (see Section III), consists of four distinguishable component types which are explained in this section, namely:

1) Simulation Coordination Component (SCC)
2) Federates
3) Observer
4) Distributed Controlling Toolkit (DistriCT)

The central component of the simulation is the *Simulation Coordination Component* (SCC) which manages the communication between the simulators and the time synchronization among them. All simulators or services, called *Federates*, register to the SCC. They specify which objects they want to be able to update and for which they want to receive updates. To support this, a common definition of all object classes and interactions exists called *Object Model Template* (OMT) [25].

Federates can be described as simulators or services connected to the SCC which have the possibility to publish data to and subscribe data from other Federates. In our considered use case two simulators exist. One of them is the *Lift Supervisor Simulator*, responsible for moving the lift supervisor on the ship deck. It uses observation paths which guarantee a free view to the cargo and the crane operator. The simulated lift supervisor chooses which path he takes on given probabilistic value. The other one is the *Physical World Simulator* (PWS), which in our example controls the environment or the crane movement. The PWS is used to provide a 3D model of the scenario and of the physical and environmental conditions. Physical effects are, for example, the collision of objects or soft body effects which are used to simulate the swinging of the crane rope. Environmental conditions are, in our case, particle effects like rain or snow. This simulator is based on the *GameKit* [26] game engine which contains *Ogre* [27] as visualization and *Bullet* [28] as physics engine. The PWS has an integrated visualization which is handy for testing purposes, manual observations, or to demonstrate the simulation. But the PWS can also be executed without graphical output to achieve a simulation speed up. More detailed descriptions of the PWS and its components can be found in Schweigert et al. [29] or Läsche et al. [25].

The *Distributed Controlling Toolkit* (DistriCT) consists of the following different program parts. *Jigger* is used to set up the simulation components on a software level. The Jigger component is divided into a client and a server part, the clients can be started on different computer systems and are available for the server. The server deploys the components, e.g., the simulators, to the clients and offers the ability to start, stop, and configure them. The *Mediator* is used to read all communicated changes via the SCC and sends data from DistriCT to the connected simulators. It allows us to listen and react to the

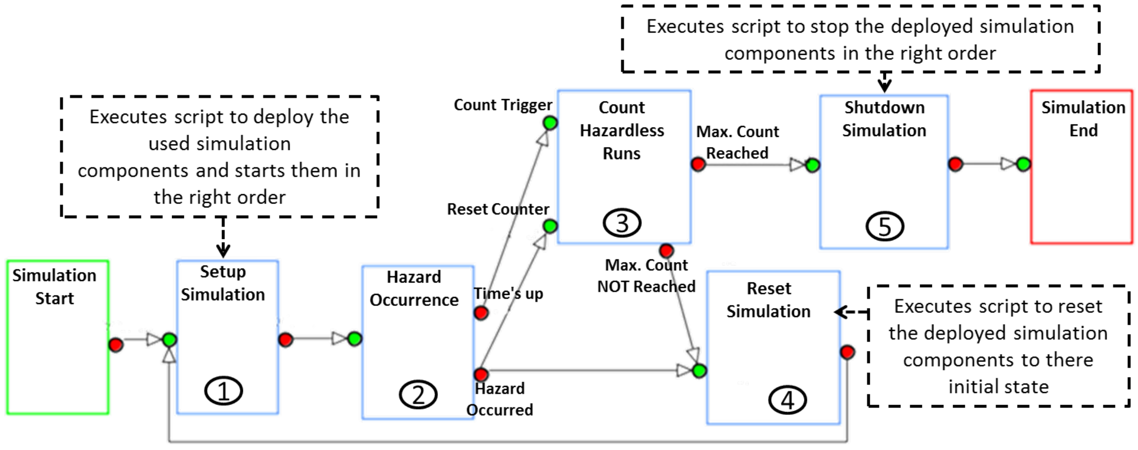

Executes script to stop the deployed simulation components in the right order
Executes script to deploy the used simulation components and starts them in the right order
Count Trigger
Count Hazardless Runs
Max. Count Reached
Shutdown Simulation
Simulation End
Reset Counter
Simulation Start
Setup Simulation
Hazard Occurrence
Time's up
Max. Count NOT Reached
Reset Simulation
Executes script to reset the deployed simulation components to there initial state
Hazard Occurred
1
2
3
4
5

Figure 3. Example of a simulation plan describing the course of simulation runs for a specific simulation scenario.

Observer when hazards or other events occur as well as instruct the other simulation participants, i.e., to reset them or to shut them down. The simulation run configuration can be described by a *Simulation Plan* which is a graph-based representation of the simulation course and can be created from a scenario description. In a simulation plan predefined nodes, like a setup node to initialize the SCC and the connected simulators, can be added. By connecting these nodes, the order of execution as well as sending necessary data from one node to another can be specified in the simulation plan. An example of a simulation plan can be found in Figure 3. In this figure, also representing our later in Section III presented use case, the SCC, the LiftSupervisor-Simulator and the PWS are created at first on a configured system; all necessary data like the mentioned OMTs is transmitted and the components are started in the right order. After that, the Mediator connects and sends the start signal to all Federates (cf. Figure 3:1). While the simulation is performed, DistriCT listens for the occurrence of hazards and the simulation time (cf. Figure 3:2). The *Logger* component writes the information about occurring failures and hazards to a file and also informs a counter (cf. Figure 3:3) if a hazard has occurred or a given maximum simulation time ran out. If a maximum number of non-hazard runs in a row was counted, the simulation is stopped (cf. Figure 3:5). The counter is set back to zero hazard-less runs if it is informed about another occurrence of a hazard. On the other hand, the simulation is reset and started again while the maximum count is not reached (cf. Figure 3:4).

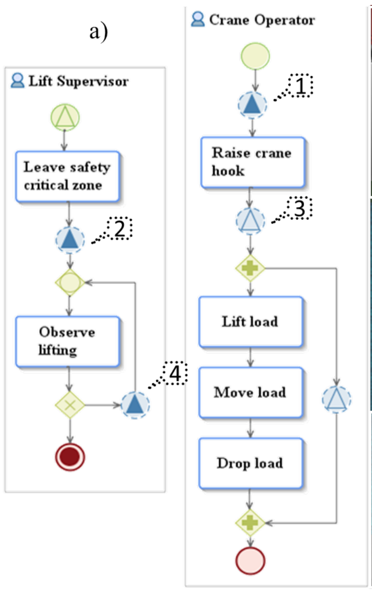
The last component introduced is *DistriCTGear* which is the central element for the other DistriCT components and allows the execution of the simulation plan and its included commands and scripts.

The introduced Observers (cf. Section II-D) are connected to the Logger component from DistriCT. They evaluate the incoming logger data for the occurrence of predefined hazards and inform connected components like DistriCT about it. Note that the Logger can also be connected to monitor-like components (Risk Monitor) which allows us to estimate a distance to a hazardous situation by a semi-automatic created heuristic and use this to trigger the saving of simulation states and running new simulations beginning at the saved state.

III. USE CASE – LOADING OPERATION

In order to illustrate our simulation-based risk assessment, we have chosen a maritime loading operation on a jack up vessel – a special ship for offshore operations, cf. Figure 4 b). A crane operator (Figure 4 b:triangle) has to lift a cargo (Figure 4 b:rectangle) with new materials and transport it to a wind turbine platform (Figure 4 b:hexagon). While there are many factors, like a bad sight or communication problems, failures might occur, leading to fatal accidents. Because of these problems during the loading operation, a person is needed to supervise the loading (Figure 4 b:circle) and point out problems to the crane operator. Therefore, the lift supervisor follows the lifted cargo to always keep it, as well as the crane operator, in his field of view (cf. Figure 4 b).

a) System Model: The described system model consists of a process model depicted in Figure 4 a and an environment model. The process model structures the use case scenario and thus tasks of the involved actors in a BPMN-like graphical representation. The use case has been adapted from literature (cf. Droste et al. [6]). It has been reduced by some preparing measures such as load preparation or hook positioning. Thus, the scenario starts with the beginning of the actual lifting operation. The crane operator signals the intended lifting of the cargo to the lift supervisor (Figure 4 a:1). The lift supervisor leaves the safety critical zone around the cargo to be lifted. Meanwhile, the crane operator raises the hook. The lift supervisor signalizes the approval of the operation (Figure 4 a:2). When the crane operator received the OK (Figure 4 a:3), the cargo then is lifted, moved to the wind turbine platform, and positioned at the target. During the operation, the lift supervisor continuously informs the crane operator about the current status from his point of view (Figure 4 a:4), whereas the crane operator steers the crane accordingly. Hence, the lift supervisor requires clear sight on the cargo, the crane operator, and potential obstacles. The scenario ends with successfully placing the cargo on the wind turbine platform. The environment model from our use case describes the following actors and physical objects and their properties. The physical objects consist of the jack up vessel, the crane on the ship and the transported cargo. The actors comprise the crane operator and the lift supervisor. The physical objects as well as the actors have a start pose (position and orientation) and a mass.



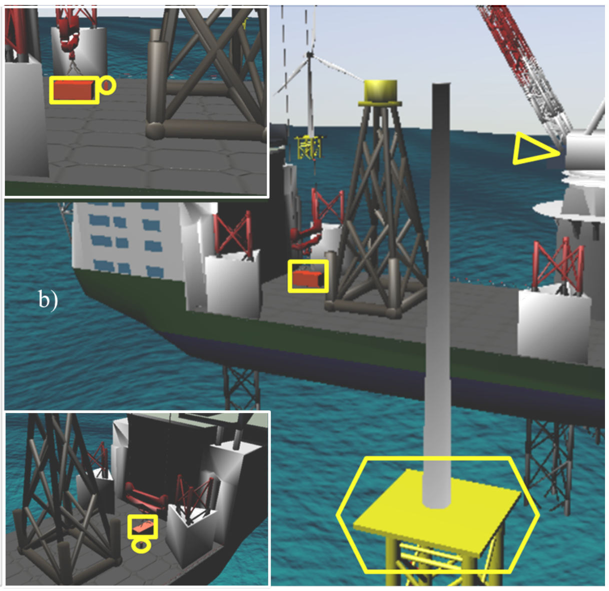


Figure 4. a) Process model of the use case including the involved actors crane operator and lift supervisor and their corresponding tasks. b) Overview of the simulation scenario from different vantage points. The lift supervisor is marked by a circle, the loading by a rectangle, the crane cockpit by a triangle, and the wind turbine platform by a hexagon.

b) Scenario Description Model: Using this system model a scenario description template is created. The jack up vessel, the crane, the transported cargo, as well as the lift supervisor and the crane operator are converted into scenario configuration elements and automatically completed by a start trigger which means that the standard property values from the environment model like the position are set at the start of a simulation run. The scenario description is then adjusted to define the simulation and simulation run ends. Therefore three exit conditions are added for the simulation end and for the simulation run ends. The simulation end is defined by a trigger listening to the current simulation run count and comparing it with a given maximum count of simulation runs. The simulation run end is defined by two triggers. The first one is a time trigger which sets the maximum simulation time to 30 seconds, the other one is an event trigger which stops the simulation run if the observer monitors a collision between the lift supervisor and the lifted cargo. From this scenario description, the simulation plan is generated (cf. Figure 3), then completed by the setup, shutdown, and reset scripts and executed by the DistriCT component.

c) Fault Tree Model: For the introduced operation, we annotated the potential hazard “lift supervisor collides with cargo” and corresponding failures, causing the hazard. These elements are depicted in the synthesized fault tree shown in Figure 6 a. The hazard occurs if a person is located below the cargo while it is dropping. Dropping of cargo can have several causes, for example it falls uncontrolled or it is intentionally dropped by the crane operator. Normally, in cases of imminent collisions the crane operator should initiate an emergency stop. If it is not initiated while the cargo is dropping, this could have several causes, e.g., technical defects which could be used to decompose one failure to more sub-failures. The fault tree depicted in Figure 6 a) represents the possible causes for a collision of the cargo with the lift supervisor. However, in the example of these failures the developer of the fault tree cannot know if there are other potential failures causing the hazard.

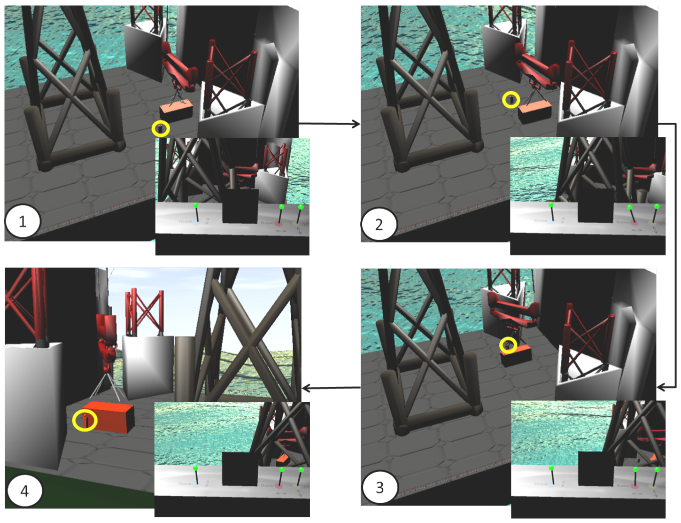


Figure 5. Collision case from load collision simulation run. **Big pictures:** Different views on the loading area short before the accident(1-3) and while the load collision hazard occurs(4). **Small pictures:** View from the crane cockpit taken at the same time as the corresponding big picture.

d) Observer: In beforehand to the first simulation run of the use case, a hazard was identified by the domain expert: A collision between cargo and lift supervisor (see top event in Figure 6 a). To avoid this situation, the domain expert analyzed the situation and identified as failure leading to that hazard that the lift supervisor is positioned below the cargo. Both situations have been formalized by the expert using HDL. The hazard “lift supervisor collides with cargo” has been described as “*distance between LiftSupervisor and Cargo equals*

0cm," i.e., the bounding boxes of the two objects overlaps. The failure "lift supervisor below cargo" has been written as "*position.x of LiftSupervisor equals position.x of Cargo and position.y of LiftSupervisor equals position.y of Cargo and position.z of Cargo is more than height of LiftSupervisor*," i.e., the position of the lift supervisor is a position below the cargo. These expressions are translated to OCL expressions during the generation of the observer. They are evaluated during execution of the simulation to indicate the occurrence of the described events.

All events that occur during each simulation run are logged. This allows the domain expert to check if the previously identified hazard occurred after the previously defined dependent failures have occurred. If this is not the case, a failure causing the hazard might have been missed or the fault tree might otherwise be faulty.

e) Simulation-based analysis: The 3D environment (see Figure 4 and Figure 5) was modeled with Blender [30]. It consists of a jack up vessel, the transported cargo, an uncompleted wind turbine, the lift supervisor and the crane operator. We completed it by a functional crane consisting of a rotatable crane base,a movable jib and a rope which can react to different physical forces (e.g., the wind). The PWS was used to load the 3D environment, handle the physical effects and control the crane. The LiftSupervisor-Simulator was used to move the lift supervisor in the 3d environment on a chosen path. Like explained in Section II-E we used the created scenario description to generate the used simulation plan shown in Figure 3.

If an error in the fault tree has been detected while observing the simulation, it has to be corrected. During the execution of the simulation plan, an observer indicated a collision between lift supervisor and cargo, although the lift supervisor was not standing below the cargo, which had been identified as a required failure.

At the beginning of the operation (cf. Figure 5:1), the lift supervisor stands next to the cargo waiting for the lifting operation to begin. While the lift supervisor reaches for his next observation point, the crane operator begins to lift the load. In the example case, the crane operator does not lift the cargo high enough and begins the rotation of the crane base too early (cf. Figure 5:2). Figure 5:3 shows the fatal development of the situation not marked as critical. The cargo is too low and moves towards the lift supervisor. The last picture, Figure 5:4, shows the actual accident. The lift supervisor is hit by the cargo. The difficulty for the crane operator to oversee the whole situation can be seen in the four smaller pictures at the corners of the big one, showing the bad sight out of the crane cockpit during the operation.

Thus, the analysis results are failures leading to the hazard that have not been considered in the synthesized fault tree, cf. Figure 6 a). The analysis discovered that the hazard occurs even if the lift supervisor is not located below the cargo (cf. Figure 6 b:1). Additionally, the results show that the rotation of the crane while cargo is raised also leads to the hazard (cf. Figure 6 b:2). These results are used to manually adapt the existing fault tree as depicted in (cf. Figure 6 b). Results are therefor used by the user to construct events such as intermediate or basic events in the fault tree as well as fault tree gates in order to logically structure simulation results in the fault tree logic. Thereby, the simulation results are already

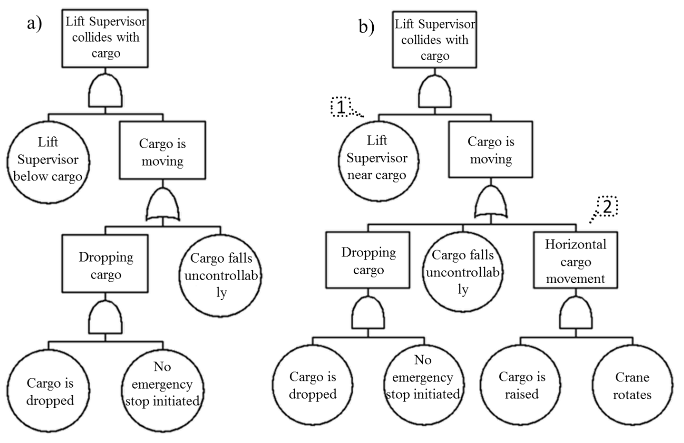

a)
Lift Supervisor collides with cargo
Lift Supervisor below cargo
Cargo is moving
Dropping cargo
Cargo falls uncontrollably
Cargo is dropped
No emergency stop initiated
b)
Lift Supervisor collides with cargo
1
Lift Supervisor near cargo
Cargo is moving
2
Dropping cargo
Cargo falls uncontrollably
Horizontal cargo movement
Cargo is dropped
No emergency stop initiated
Cargo is raised
Crane rotates

Figure 6. Fault trees of the load collision scenario. a) Exemplary fault tree before simulation analysis b) Adapted fault tree by simulation-analysis results. 1: Adapted existing fault tree element 2: New failure combination discovered by simulation

clustered as failure combinations which already propose a logical structure of found failures based on the simulation runs.

Using this setup, we tested our framework by recording the number of simulations needed to find the missing failure which could lead to the overall hazard that the cargo hits the lift supervisor. Empirically, averaging across 50 iterations, we found that 6.43 simulations were needed to identify the missing hazard. Such few number of simulations, however, will in general not suffice in other situations in which the probability of an overall hazardous event is much lower. In contrast, the high probability here was chosen for demonstration purposes and therefore does not require the use specialized simulation methods to detect rare events.

IV. Conclusion and Future Work

In this paper, we have presented a simulation-based approach to identify potentially missing failures of a given fault tree that represents a critical scenario in the maritime domain. Importantly, the presented method comes with a confidence statement about the statistical certainty about the existence of failures which have not been identified by the simulation. To provide sufficient evidence for a complete characterization of potential failures, a large number of simulation runs are required. Although this is a well-known problem in statistical model checking, setting up appropriate rare event simulations remain subject to future research. However, these confidence statements assume the correctness of the underlying model for simulation. The check for failures should be seen as a means for finding additional critical situations and therefore acts as a completeness check of a previous risk analysis. Nevertheless, having a simulation model at hand also allows for further analysis methods, which unlike the presented method can also access risks during an (offshore) operation by initializing the simulation with the current status and specifically search for potential risks in the near future conditional on the current situation. Again, to bound the number of necessary simulation runs and acquire sufficient information about the current level of risk, techniques from rare event simulation can be used. In particular, importance splitting (cf. Lagnoux [21]) can be used to guide the simulation to potentially more risky situations.

To this end, an artificial risk level has to be defined, which heuristically estimates the risk level and guides the simulation. Defining such heuristic and integrate a corresponding importance sampling technique into the presented approach will therefore be the focus of future research.

ACKNOWLEDGMENTS

This work was partially supported by the *European Regional Development Fund* (ERDF) within the project *Safe Offshore Operations* (SOOP) [31].

REFERENCES

[1] K. E. Thomsen, Offshore Wind: A Comprehensive Guide to Successful Offshore Wind Farm Installation. Amsterdam: Elsevier Ltd, Oxford, 2011.

[2] Guidelines for Onshore and Offshore Wind Farms – Health & Safety in the Wind Energy Industry Sector, RenewableUK, London, 2010.

[3] Guidelines for Onshore and Offshore Wind Farms – Health & Safety in the Wind Energy Industry Sector, RenewableUK, London, 2013.

[4] W. Vesely, Fault Tree Handbook with Aerospace Applications. Washington DC: Nasa, 2002.

[5] S. Bangsow, Manufacturing Simulation with Plant Simulation and SimTalk. Springer Science & Business Media, 2010.

[6] R. Droste, C. Läsche, C. Sobiech, E. Böde, and A. Hahn, "Model-based risk assessment supporting development of hse plans for safe offshore operations." in Formal Methods for Industrial Critical Systems, ser. Lecture Notes in Computer Science, M. Stoelinga and R. Pinger, Eds., vol. 7612. Springer Berlin / Heidelberg, 2012, pp. 146–161.

[7] OMG, Business Process Model and Notation (BPMN) Version 2.0, 01 2011.

[8] ——, Unified Modeling Language (UML) Version 2.4.1, 08 2011.

[9] C. Läsche, R. Droste, J. Pinkowski, S. Gerwinn, and A. Hahn, "Model-based risk assessment of offshore operations," in Proceedings 33rd International Conference on Ocean, Offshore and Arctic Engineering. ASME, 2014, proceedings available CD-ROM.

[10] J. E. Vinnem, Offshore Risk Assessment. London: Springer Series in Reliability Engineering, 2007.

[11] M. M. Lavasani, J. Wang, Z. Yang, and J. Finlay, "Application of fuzzy fault tree analysis on oil and gas offshore pipelines," Int. J. Mar. Sci. Eng, vol. 1, no. 1, 2011, pp. 29–42.

[12] I. S. Sutton, Offshore safety management: implementing a SEMS program. William Andrew Publishing, 2011.

[13] M. L. McKelvin Jr, G. Eirea, C. Pinello, S. Kanajan, and A. L. Sangiovanni-Vincentelli, "A formal approach to fault tree synthesis for the analysis of distributed fault tolerant systems," in Proceedings of the 5th ACM international conference on Embedded software. ACM, 2005, pp. 237–246.

[14] F. Tajarrod and G. Latif-Shabgahi, "A novel methodology for synthesis of fault trees from matlab-simulink model," World Academy of Science, Engineering and Technology, vol. 41, 2008, pp. 630–636.

[15] B. Chen, "Improving processes using static analysis techniques," Ph.D. dissertation, University of Massachusetts Amherst, 2011.

[16] G. J. Pai and J. B. Dugan, "Automatic synthesis of dynamic fault trees from uml system models," in Software Reliability Engineering, 2002. ISSRE 2003. Proceedings. 13th International Symposium on. IEEE, 2002, pp. 243–254.

[17] G. Schellhorn, A. Thums, W. Reif et al., "Formal fault tree semantics," in Proceedings of The Sixth World Conference on Integrated Design & Process Technology, Pasadena, CA, 2002, pp. 1–8.

[18] F. Ortmeier and G. Schellhorn, "Formal fault tree analysis-practical experiences," Electronic Notes in Theoretical Computer Science, vol. 185, 2007, pp. 139–151.

[19] OMG, Documents Associated With Object Constraint Language, Version 2.0, 05 2006.

[20] International Electrotechnical Commission , IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems, 2010.

[21] A. Lagnoux, "Rare event simulation," Probability in the Engineering and Informational Sciences, vol. 20, 1 2006, pp. 45–66.

[22] D. Reijsbergen, P.-T. de Boer, W. Scheinhardt, and B. Haverkort, "Rare event simulation for highly dependable systems with fast repairs," in Proceedings of the Seventh International Conference on Quantitative Evaluation of SysTems, QEST 2010. Los Alamitos: IEEE Press, July 2010, pp. 251–260.

[23] P. Zuliani, C. Baier, and E. Clarke, "Rare-event verification for stochastic hybrid systems," in Hybrid Systems: Computation and Control, 2012, pp. 217–225.

[24] C. Jegourel, A. Legay, and S. Sedwards, "Importance splitting for statistical model checking rare properties," in Computer Aided Verification, 2013, pp. 1–16.

[25] C. Läsche, V. Gollücke, and A. Hahn, "Using an hla simulation environment for safety concept verification of offshore operations." in ECMS, W. Rekdalsbakken, R. T. Bye, and H. Zhang, Eds. European Council for Modeling and Simulation, 2013, pp. 156–162.

[26] gamekit – A cross-platform 3D game engine using Ogre or Irrlicht and Bullet for Windows, Linux, Mac, Android and iPhone. [retrieved 07, 2014]. [Online]. Available: https://code.google.com/p/gamekit/

[27] OGRE – Open Source 3D Graphics Engine. [retrieved 07, 2014]. [Online]. Available: http://www.ogre3d.org/

[28] Bullet – Real-Time Physics Simulation. [retrieved 07, 2014]. [Online]. Available: http://bulletphysics.org/

[29] S. Schweigert, R. Droste, and A. Hahn, "Multi-Agenten basierte 3D Simulation für die Evaluierung von Offshore Operationen," in Go-3D 2012 – »Computergraphik für die Praxis«, 2012, pp. 105–126.

[30] blender.org – Home of the Blender project. [retrieved 07, 2014]. [Online]. Available: http://www.blender.org/

[31] soop.offis.de - Sichere Offshore-Operationen. [retrieved 07, 2014]. [Online]. Available: http://soop.offis.de/