A Model-Based Approach to Support Safety-Related Decisions in the Petroleum Domain

Leonardo Montecchi[1,2], Atle Refsdal[3], Paolo Lollini[1,2], Andrea Bondavalli[1,2]

[1] University of Firenze – Firenze, Italy – {lmontecchi,lollini,bondavalli}@unifi.it
[2] Consorzio Interuniversitario Nazionale per l'Informatica (CINI), University of Firenze – Firenze, Italy
[3] SINTEF – Oslo, Norway – atle.refsdal@sintef.no

***Abstract*—Accidents on petroleum installations can have huge consequences; to mitigate the risk, a number of safety barriers are devised. Faults and unexpected events may cause barriers to temporarily deviate from their nominal state. For safety reasons, a work permit process is in place: decision makers accept or reject work permits based on the current state of barriers. However, this is difficult to estimate, as it depends on a multitude of physical, technical and human factors. Information obtained from different sources needs to be aggregated by humans, typically within a limited amount of time. In this paper we propose an approach to provide an automated decision support to the work permit system, which consists in the evaluation of quantitative measures of the risk associated with the execution of work. The approach relies on state-based stochastic models, which can be automatically composed based on the work permit to be examined.**

Keywords—safety analysis; barriers; petroleum; risk evaluation; model-based.

I. INTRODUCTION

Petroleum installations are complex socio-technical systems where accidents can have major impact on life, health and the environment; furthermore, they can lead to huge economic losses. On the other hand, unnecessary interruption of operations would have a strong economic impact as well. For this reason, petroleum installations are subject to frequent work activities, to be undertaken during normal system operation. Work activities include maintenance operations, periodic verification of equipment, but also routine work operations. Frequently, the execution of an activity is subject to safety threats, for example when handling hazardous materials (e.g., flammable materials).

To mitigate the risk of accidents, a number of countermeasures, called *safety barriers*, are devised and implemented across the entire system. Collectively, safety barriers constitute the "barrier system", a complex (socio-technical) system whose behavior depends on a multitude of physical, technical, and human factors. In fact, barriers may consist of physical components (e.g., gas detectors), software, procedures, trained humans, and basically everything that may contribute to risk reduction.

Among these, a central role is played by the work permit process [3], which prescribes that work on installations can be carried out only if a specific written authorization (*work permit*) has been released. Deciding whether to release or not a certain work permit is a complex decision, which requires taking into account different aspects, including the kind of work, possible conflicting work in the same area, as well as the current state of the safety barriers that are relevant for that work. Furthermore, barriers are subject to faults, and understanding to which extents they are able to perform as intended is fundamental. Such decisions are taken by humans, and often need to be taken within a limited amount of time. Supporting tools for this process are so far limited, which means that decision makers have to aggregate the information gathered from different sources and conclude, based on their expertise, whether authorizing a certain work would be "sufficiently" safe or not.

In this paper we propose a methodology to support the work permit decisions process in the petroleum domain. The approach we propose is to compute quantitative measures oriented to the evaluation of the risk associated with the execution of a certain work, using stochastic state-based models. Developing a decision support system will allow operators to have aggregate and *objective* information at their disposal, thus being able to take informed decisions, and document the motivations behind them. It should be noted that we are addressing a different problem with respect to safety cases, which serve a different purpose, and should hold for very long time. Conversely, the problem we address is evaluating the short-term risk associated with executing specific actions on installations.

The paper is organized as follows. The problem we tackle is detailed in Section II, in which we introduce the work permit system and current challenges. Related work is discussed in Section III. Our model-based approach and the information it requires are illustrated in Section IV; the defined model library is presented in details in Section V. The application to a simple case study is then reported in Section VI, while in Section VII we discuss on the plans for automation and integration of the proposed steps. Finally, conclusions are drawn in Section VIII.

II. SAFETY PROCEDURES IN THE PETROLEUM DOMAIN

A. The Work Permit System

A central task to ensure safety on petroleum installations is to coordinate work, in order to avoid conflicts and ensure that potentially risky work is not initiated unless appropriate countermeasures, called safety barriers, are in place. For example, if in a certain area gas detectors are in a degraded state due to overdue maintenance, then hot work such as welding should not be allowed in that area, unless appropriate compensating

measures are in place. This is enforced by the "work permit" system, which should ensure safe execution and coordination of work on installations [3].

Workers that need to perform non-routine work on an installation have to apply for a *work permit* (WP), by filling out a standardized form [3]. Every 12th hour, decision makers hold a meeting to go through all incoming WP applications for the next 12 hours, and decide which ones to release (accept) and which ones to reject. The number of WP applications can be high, meaning that the time available for each decision is limited. Making the right decision requires a good understanding of the current state of safety barrier systems, which may consist of many different parts and components, possibly having complex interactions.

Which system components are involved, and how they contribute to the safety of work execution depends on a combination of: i) the cyber and physical architecture of the system, ii) the specifics of the work to be authorized, described in the WP, and iii) the current state of system components. Among all the aspects that decision makers have to take into account, information about deviations of safety-related components from their nominal state are of primary importance. This includes, for example, components that are overdue for periodic maintenance, errors detected but not yet fixed, and so on. This kind of information is typically collected during system operation, and stored in a "deviations database". To take a properly informed decision, decision makers need to identify, extract and aggregate all the relevant information in order to assess the overall state of the barriers.

This is a very difficult task, which is made even more difficult by the impact of subjectivity and possible human errors. For this reason, automated decision support, integrated into a computer-based WP system, has been theorized as a welcome improvement to the WP process, as illustrated in Fig. 1. The process starts with the applicant submitting the WP application to the WP system; the application is then presented to the decision maker at the appropriate time. Based on the specific information provided in the application, the WP system checks the status of relevant safety barriers, aggregates the obtained information and provides, along with the WP, some quantitative metrics related to the risk of executing that work. Optionally, if the system estimates that the state of barriers is unsatisfactory with respect to some predefined threshold, then it issues a warning accompanied by an explanation, which is notified to the decision maker.

It is then up to the decision maker to finally decide whether to release or reject the WP. The decision is issued to the WP system and forwarded to the applicant, as illustrated by the two alternatives of the "alt" construct at the bottom of the diagram. We emphasize that the decision maker is ultimately responsible for the decision and will typically consider not only the warning (or absence of warning) from the WP system, but also additional knowledge that may not be available to the automated decision support system, e.g., the expertise of workers.

B. Challenges and Success Criteria

Realizing a complete decision support system as described above requires a number of challenges and gaps to be addressed. In the following we summarize the main challenges that we identified, and discuss the ones that we aim to address with our proposal.

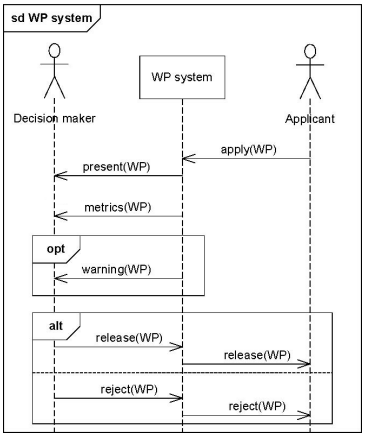

Fig. 1. WP system with decision support.

The first challenge consists in being able to accurately perform run-time monitoring on all the safety-relevant aspects of a complex system like a petroleum installation. This includes keeping all the data in a well-organized and efficient database, and ensuring that stored information does not become outdated. Deciding which variables to monitor, and the acquisition interval is also an important challenge. We do not address this aspect, and assume (as it is the case) that monitoring facilities are put in place by the owner of the infrastructure.

The kind of input that should be processed by the WP system introduces additional complexity. Although standard forms have been established [3], a work permit application is to a large degree constituted by free text, which should somehow be interpreted by the WP system. This aspect is also not addressed here: we assume that the information we need is available in a structured way in the WP application. Such information can be extracted via language processing techniques, or simply by using a structured format for WP applications. We believe that this will happen as operators switch from paper-based to computer-based work permit systems. The work in this paper was inspired by cooperation with software providers currently pioneering this development.

The aggregation of the information contained in the database, and in the WP application, is also a great challenge. Synthetic indicators need to be devised, so that the decision maker has an immediate understanding of the current state of the system. The decision maker should be able to understand the semantic behind the aggregated metrics, but he/she should be alleviated from understanding how they have been computed, which may involve the application of complex techniques. The approach we introduce in this paper addresses this challenge using quantitative safety metrics and state-based stochastic models.

A final aspect to be considered is the high variability of different scenarios that the WP system should be able to address. The impact of faults, their propagation, and the way in which they impair the safety barriers is very variable, based on the kind of WP which is being requested. Different WPs may ap-

ply to different areas of the system and different kind of works; different conditions of the system may further introduce variability. The approach proposed in this paper addresses this problem by defining a set of template SAN [11] submodels that can be combined in different ways to automatically evaluate different scenarios.

III. Related Work

Approaches for the evaluation of safety properties in the petroleum domain have been introduced in the literature. However, none of them have been universally adopted by the industry.

The BORA-Release method [4] is designed specifically for barrier and operational risk analysis of hydrocarbon releases. It addresses a full risk analysis process and therefore it has a broader scope than our approach. One of the steps of the process consists of *modeling the performance of safety barriers*, which is performed using fault trees. With respect to fault-trees, state-based stochastic models like those used in this paper allow more detailed behaviors to be modeled (e.g., failure propagation, time-related aspects). At the same time, we believe that our approach provides an alternate solution for the modeling of safety barriers that could be plugged into the BORA-Release method.

Røed et al. [5] propose an interesting approach for the analysis of risk on petroleum installations. Their approach allows some parts of the risk assessment to be addressed using fault trees, while other parts are addressed using Bayesian belief networks. Hybrid casual logic is then applied to analyze the barriers in terms of probability of barrier failure. However, there are fundamental differences in the way barriers are modeled; for example, our approach is able to take into account the temporal dimension. Furthermore the authors state that their technique is resource intensive.

Other approaches, like ours, are geared towards a model-driven approach. The work in [9] introduces an approach for the safety analysis of socio-technical systems, based on failure logic analysis techniques. The approach is applied to a case study in the petroleum domain. The authors of [10] introduce an approach for compositional risk modeling, through a notion of risk model encapsulation, for which internal details of a risk model are hidden. This is achieved by defining a risk model interface that contains all and only the information that is needed for composing the individual risk models to derive the overall risk picture. However, the focus of those papers is mainly on the modeling aspects; also, they do not address techniques for quantifying the risk, but only perform qualitative analyses.

The approach we propose in this paper is based on stochastic state-based models. State-based stochastic models have been widely adopted in the literature on dependability and performance analysis [7][25], also in combination with simulative and experimental measurement approaches [23]. Traditionally, such techniques have been applied for the analysis of specific problems, in which the system architecture was almost fixed. Successively, as system became more complex, the need for a different perspective arose, leading to techniques to automatically derive analysis models from more abstract system descriptions. This research direction is still active, today mostly focusing on the UML language [8].

In an effort to improve the extent to which state-based models can be reused, much like ordinary "components", different authors have recognized the benefits of applying modularization and "separation of concerns" [12] to the construction of state-based stochastic models. In such approaches (e.g., see [13][14][15][22]) the overall analysis model is built out of submodels addressing specific aspects of the systems, and having well-defined interfaces. Such submodels are then composed following predefined rules based on the actual scenario to be represented. The authors of [16] further elaborated on this concept, defining an approach in which SAN models extracted from model libraries are composed together, by model-transformations, based on semi-formal specifications of the scenario to be represented. The approach we propose in this paper falls in this category, as it is based on a library of models that can be automatically assembled to represent different system configurations. Such versatility helps in coping with the complexity of the safety barriers and the variability of WP applications.

IV. Model-Based Approach for WP Processing

A. The Methodology

State-based stochastic models are a valuable tool for constructing the "warning" messages of Fig. 1: they are able to take into account complex interactions and dependencies, and aspects related to time and probabilistic behavior. By applying model-based evaluation [7] on a model of the safety barriers that are relevant to the work permit under examination, specific safety-related metric could be evaluated, and a warning message generated if they fall below a predefined threshold.

However, we need to construct the model based on the information present in the WP. The approach we propose assumes to have at its disposal: i) a library of architectural models of barriers, specifying components and interconnections, as well as the unwanted events that need to be avoided; ii) a library of "template" SAN models, specifically created for the WP problem; and iii) a database in which information on the current state of the installation is stored. As mentioned in Section II, the latter is typically in use in current practice already.

The high-level view of the proposed approach is illustrated in Fig. 2. The workflow takes as input a WP application, and provides as output a decision on whether the WP should be released (accepted) or rejected. A preliminary step consists in analyzing the WP application, in order to retrieve the information that is relevant to the decision process. This includes identifying the relevant characteristics of the WP, e.g., the kind of work and the areas in which it will be undertaken, and consequently the safety barriers that are relevant to that WP application. The effort required to perform this activity depends on the actual representation of the WP form that is adopted by the company. In case of a strongly structured representation (e.g., semi-formal languages, XML, etc.), information can be directly retrieved from the WP. If it is not the case, a *WP Analysis* step is performed to extract the required information. In this paper we assume that the needed information, consisting in the *Rele-*

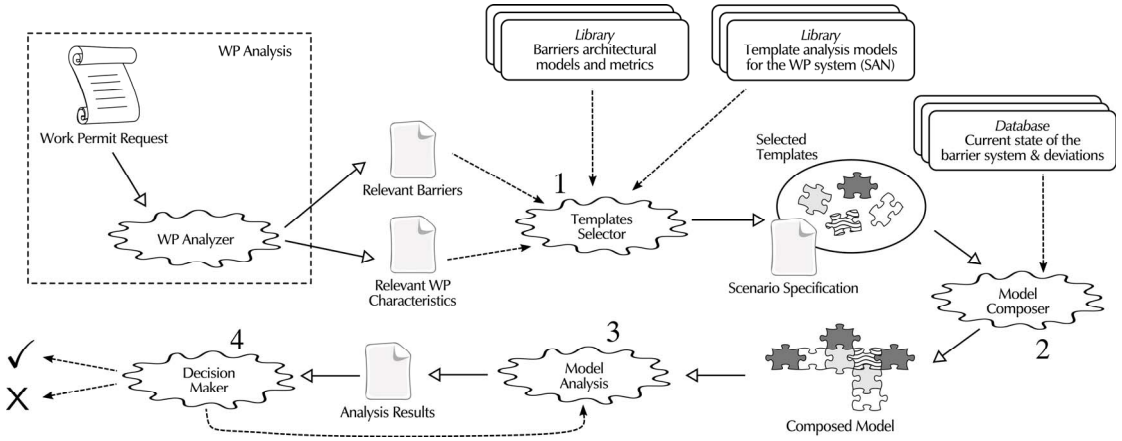

Fig. 2. Workflow of the proposed methodology for building a decision support WP system.

vant Barriers and the *Relevant WP Characteristics* has already been extracted from WPs.

Following the application, the information extracted from the WP is processed by the *Templates Selector* module (1), which, based on the architectural models of the involved barriers and on the specificities of the WP, identifies the template models that need to be retrieved from the model library, and how they should be interconnected. The complete composite model for evaluating the WP request of interest is then assembled by the *Model Composer* module (2).

The next step is to evaluate the model and obtain the numerical results for the metrics of interest (3). Such results are then presented, possibly accompanied by warning messages, to the decision maker (4), which then ultimately decides whether to release the WP or not. Before taking the final decision, he/she can request additional evaluations of the model, for example to perform sensitivity analysis on some key parameters or to perform "what-if" analysis with respect to conditions that are judged to be particularly critical.

In this paper we focus on the key steps 1 and 2 of the workflow, and then provide an application to a simple use case, which also shows a possible execution of steps 3 and 4.

B. Barriers Architectural Models

As a prerequisite to the execution of step 1, we assumed to have at our disposal a library of architectural models of barriers. These models are created and maintained by collaboration between petroleum domain experts and safety experts. Each of those models describes one or more events to be avoided and/or detected, the chain of events that may cause them, and possible barriers that prevent such events from propagating. Ideally, those models need to be updated only rarely; in such occasions, their consistency and adherence to reality are thoroughly verified.

The actual language used to model barriers is a choice that is related to technical details, and to preferences of individual companies. We do not pose restriction on the language that is adopted for modeling barriers: at this level we are only interested in the information that should be available, which consists in the following:

- The *events* of interest for that barrier (i.e., events that the barrier is meant to prevent or detect). Events have an *occurrence* probability distribution.
- The *components* that are part of the barrier. Component is here intended in a broader sense, also considering human and organizational entities as in [9]. Components are affected by *internal faults*, which occur with a given *fault occurrence* probability distribution, and by *external faults*, which are caused by other components [1]. A component can have one or more *failure modes*, which are caused by different combination of internal and external faults.
- How component failures propagate, i.e., how the failure of a component affects components that depend on it or communicate with it. More precisely, which *failure modes* of a component induce which *external faults* of components that depend or communicate with it. Possibly, propagation may occur after some *delay*.
- How *events propagate/escalate*, i.e., how they combine to form a new event. Propagation of an event may be *immediate*, if the event is a direct consequence of a combination of events, or after a time specified by an *occurrence* probability distribution.
- Components able to *block* the propagation of events, if any. Some components of the barrier, when working correctly, are able to block the propagation of events. Some of them are *consumable*, meaning that can be used only once (or a limited number of times). It is the case for example of fire extinguishers.
- Components able to *detect* events, if any. Some components of the barrier may be able to detect events occurring in the installation. The extent to which they are able to do this depends on their working conditions. We distinguish between *timely* detection of events, and *late* detection of events, i.e., detection that occurs after some predefined time constraint.
- The set of metrics of interest for that barrier. Possible metrics are related to the likelihood of occurrence of potential incidents: i) the probability of occurrence of an event; ii) the probability of not detecting an event on

time, iii) the probability of not detecting an event at all; iv) the probability that a given combination of component failures occur. Each barrier model specifies the exact metrics that are relevant for that barrier.

Such information needs to be represented in a structured way, and stored in a library, in order to be processed by the *Template Selector* module. Suitable languages include UML [21], but also simpler representations like XML documents.

C. Work Permit Request

We then need to identify which of the information present in a WP application is of interest for our approach. We based our inspection on the standardized WP forms provided as appendix to [3]. There exist two kinds of WP forms: Level 1 and Level 2 forms, based on the risk associated to the work to be executed. Where not specified otherwise, the following list applies to both WP levels.

Date and time. Basic information of the WP request contains the date and time of execution of the work, in the fields "Date", "From hour" and "to hour". This information is important in order to retrieve up to date parameters from the database, and to know the interval of time in the future for which metrics should be predicted.

Kind of work. The kind of work that will be performed is specified by the "Work description" field. For a Level 1 WP, additional checkboxes are used to precisely state if specific kinds of dangerous work (e.g., pressure testing) will be performed. The kind of work contributes to the identification of risks associated to its execution, and consequently of the safety barriers that need to be in operation.

Location. The location of the work is precisely identified by a number of fields: "Installation", "Location/module", "Deck" and "Zone". The precise location allows risks associated to the location to be identified (e.g., failed components), and also to identify possible conflicting work to include in the evaluation.

Risks. A specific "Identified risks" section of a WP application contains a description of the risks associated with the work. This includes a description of the hazards that may be generated by the work, the accidents that may occur, and which measures should be implemented in order to mitigate the identified risk. The information in this field is used to identify the events that must be avoided, i.e., those whose probability of occurrence will be estimated.

Required operations and safety measures. Based on the kind of work and associated risks, the applicant also fills a section to indicate which are, in his/her experience, the required operations and safety measures to be established. This includes, for example, preventing the release of oil/gas in the area, the use of fire extinguishers, and the periodic measurement of gas levels. The information in this field contributes to the selection of barriers to be included in the analysis model.

Isolation of safety systems. In this section it should be specified if the execution of the work requires disconnecting some safety system, in which area of the installation, and which are the needed countermeasures. This constitutes important information not only in relation to the WP under examination, but also in relation to other work that is concurrently ongoing or planned in the same area.

D. Data Repository

The last element that we use as source of information is a data repository where information about the installation and its components is stored, obtained from monitoring, historical data, or datasheets. In a typical installation, the information that can be stored in such a repository is huge. In this paper we assume to know at least the following information, which is mostly available in current practice already:

- Failure rates or failure distributions of components, obtained from datasheets or historical data.
- Relative occurrence of different failure modes of components, obtained from datasheets or historical data.
- Occurrence rates or distributions of events, obtained from historical data or experts' judgment.
- Time of last repair/installation/verification of components.

V. Analysis Models Templates

The above information should drive the construction of the stochastic analysis model from a set of reusable submodels having predefined interfaces and parameters. Such submodels, created using the SAN formalisms, are organized in a library, from which – on demand – they are instantiated multiple times and connected together to model the scenario concerning the WP under examination. Templates constituting the library, and how they are interconnected, are described in this section.

A. Template Models

The library includes 8 templates: *EventOccurrence*, *EventRecovery*, *ConditionChecker*, *GenericComponentGenericComponent*, *Detector*, *ConsumableBarrier*, *Disabler*, *Timeline.* Each of these templates models a key aspect among those introduced in the previous sections; when combined together, the behavior of whole barriers can be represented.

The adopted formalism is Stochastic Activity Networks (SAN) [11], an extension of Stochastic Petri Nets (SPNs) [2]: *places* (represented as circles) can contain *tokens*; the number of tokens in all places is the state of the model. Tokens are added or removed by the firing of *activities* (represented as vertical bars), which may be immediate or timed. *Input gates* and *output gates* (represented as triangles) allow complex preconditions and consequences to be specified for the execution of activities. Further information on the SAN formalism can be found in [11]. When it is not specified otherwise, we do not set constraints on the probability distribution of timed activities. Interface places, i.e., those that are meant to be shared with the other subnets, are highlighted with a dashed box in the following figures.

1) EventOccurrence

The *EventOccurrence* subnet (Fig. 3) models the occurrence of a generic event related to the platform. For example,

this subnet can be used to model dangerous events like gas leakage or modifications of weather conditions.

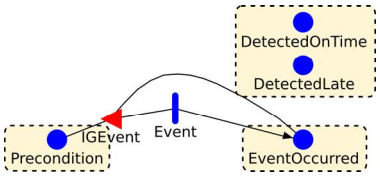

Fig. 3. EventOccurrence subnet.

The preconditions for the occurrence of the event are represented by the *Precondition* place, which contains a token only if the needed preconditions are not satisfied. The actual occurrence of the event is modeled by the firing of the *Event* activity; the occurrence of the event normally does not modify its preconditions, i.e., tokens are not removed from the *Precondition* place. A token is added to the *EventOccurred* place.

Two additional places, *DetectedOnTime* and *DetectedLate* are associated with each event, and are used to keep track of whether an event has been detected or not, and if it has been detected on time or late (with respect to an appropriate deadline for its detection).

2) EventRecovery

Recovery from the occurrence of an event is modeled by the *EventRecovery* subnet (Fig. 4). In general, this means that the immediate effects caused by the event have been removed. For example, the gas leakage has been stopped, or a good weather condition was restored.

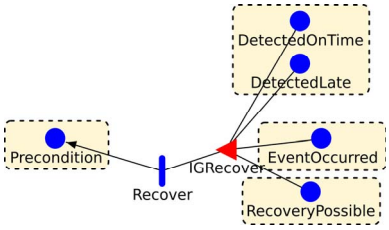

Fig. 4. EventRecovery subnet.

Also the recovery requires the existence of a precondition, which is modeled by the *RecoveryPossible* place. Only when such place contains a token the *Recover* activity is enabled. When the *Recover* activity fires, all tokens in places *EventOccurred*, *DetectedOnTime*, and *DetectedLate* are removed.

It should be noted that the precondition for the execution of a recovery event may also be the occurrence of another event. In this case, the *EventOccurred* place of the second event is shared with the *RecoveryPossible* place of the event for which recovery is being modeled.

3) GenericComponent

The subnet for modeling a generic component is depicted in (Fig. 5). A component may be affected by a certain number of different failure modes, which may be caused by internal faults, external faults, or a combination of both.

The occurrence (and activation) of an internal fault is represented by the *FaultOccurrence* activity. With a certain probability, the activation of the fault may generate different errors in the component, each one represented by an *ErrorX* place. In the example in the figure three errors are shown; the actual number depends on the concrete component to be modeled.

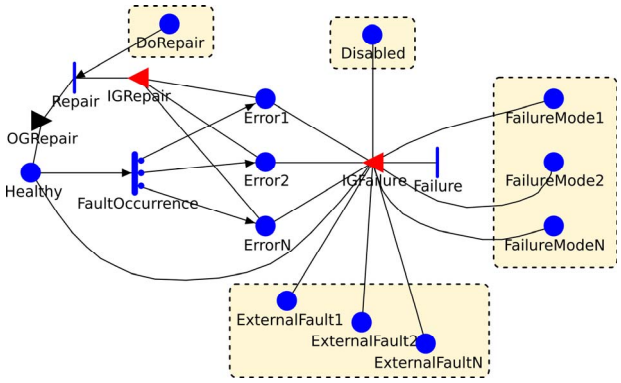

Fig. 5. GenericComponent subnet.

Failure modes of the component are modeled by a set of *FailureModeX* places, which contain a token if that specific failure mode has occurred, and are the interface of the subnet with the other models. The component may be affected by external faults, represented by the *ExternalFaultX* interface places. These places are shared with *FailureMode* places of other components, or with *EventOccurred* places of events that are external faults for the component (e.g., the occurrence of an electric shock). As a particular kind of external fault, the *Disabled* place contains a token when the component is explicitly disabled during the work, e.g., for work that includes the "isolation of safety systems" [3].

The occurrence of component failures is given by a combination of external faults, errors, and the disabling event. The condition is encoded in the *IGFailure* input gate associated with the *Failure* activity. The firing of this activity ensures that the marking of the *FailureModeX* places is always consistent with the current state of the component. We assume that the effect of errors and external fault is immediate: we do not consider propagation delays. This is mainly due to the fact that information on the system is limited, and such delays would be difficult to estimate. Still, in case a delay needs to be modeled, it can be represented using the *EventOccurence* subnet.

Repair of the component is modeled by the *Repair* activity, which is enabled and fires when a token is in *DoRepair* place. To ensure a correct modeling of the repair, the *FaultOccurrence* activity is reactivated (i.e., a new value is sampled from its distribution) when a token is in *DoRepair*.

4) ConditionChecker

For several reasons it may be necessary to test specific conditions on the state of the overall system. For example, a precondition for the execution of an event may be satisfied only if a specific combination of other events has occurred.

This kind of check is performed by means of the *ConditionChecker* subnet (Fig. 6). This subnet has a set of places as input, each of them representing one of the basic conditions to be checked (e.g., the occurrence of some event, or the failure of a component). The place *Condition* holds one token if the condition to be checked is true, otherwise it is empty. The objec-

tive of this subnet is to ensure that the number of tokens in the *Condition* place is always consistent with the result of evaluating the specified condition on the *InputX* places.

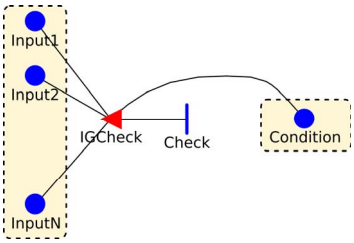

Fig. 6. ConditionChecker subnet.

The condition to be checked is encoded in the *IGCheck* input gate, which ensures that the number of token in place *Condition* is consistent with the state of the places considered as input of the subnet. If it is not the case, then *Check* fires and the execution of the *IGCheck* gate reestablishes the consistency.

5) Detector

Event detection is modeled using the *Detector* subnet (Fig. 7), which interfaces with the event that it is meant to detect, through the interface places *EventOccurred*, *DetectedOnTime*, and *DetectedLate*.

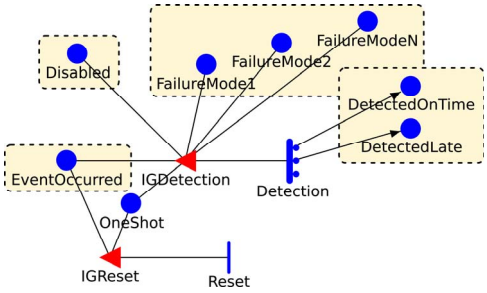

Fig. 7. Detector subnet.

The actual detection is modeled by the *Detection* activity, which may have three outcomes: the event is detected within a predefined time bound; the event is detected late; the event is not detected at all. The relative probabilities of the three corresponding cases are a parameter of the template. Place *OneShot* ensures that the activity fires only once per event occurrence. When the reference event is cleared (i.e., place *EventOccurred* becomes empty), then a token is restored in the *OneShot* place by the firing of the *Reset* activity.

In case the detection activity relies on some specific component of the barrier system, then the subnet is connected to the corresponding *GenericComponent* subnet, by means of the *Disabled* interface place, and the *FailureModeX* places. The content of these places may influence the firing of the *Detection* activity and/or the relative probabilities of its cases.

6) ConsumableBarrier

Some safety barriers may be consumable, i.e., they may be used only once and need to be replaced after their use. It is the case for example of fire extinguishers. This aspect is modeled by the *ConsumableBarrier* model (Fig. 8). The model contains two instantaneous activities, *Block*, which represents the application of the consumable barrier, and *Propagate*, which represents the propagation of the incoming event, due to the depletion of the consumable barrier items.

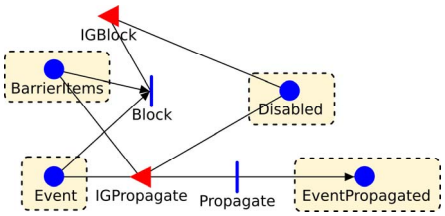

Fig. 8. ConsumableBarrier subnet.

The number of consumable items is stored in place *BarrierItems*, and it is progressively diminished every time the *Block* activity fires. The *Disabled* place represents the disabling of this barrier. In case the consumable barrier explicitly refers to a component of the system, the *BarrierItems* is connected with the *Healthy* place of the corresponding subnet.

7) Disabler

Disabling of a component is modeled using the *Disabler* subnet (Fig. 9). The *Disabled* place is shared with the corresponding place in the subnet to be disabled, while *DisablingStart* and *DisablingEnd* are two interface places that signal the beginning and the end of the disabling operation, respectively. Such places are connected with *EventOccurred* places of events that trigger the disabling procedure, or with places of a *Timeline* model (described below). A token in place *DisablingStart* triggers the *Disable* activity, while a token in *DisablingEnd* re-enables the component through the *Enable* activity.

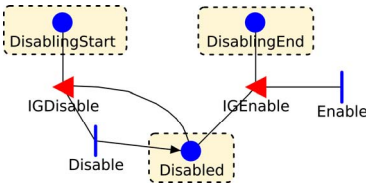

Fig. 9. Disabler subnet.

8) Timeline

While some events occur spontaneously or as consequences of other events, others are known in advance, e.g., because they are part of the plan described in the work permit, or because they represent some past event that need to be included in the model. Such events thus follow a precise sequence and timing. This aspect is modeled by a *Timeline* subnet, which describes a sequence of known phases.

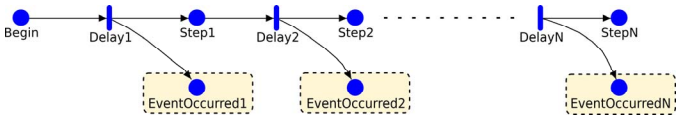

Fig. 10. Timeline subnet.

An example of such model is depicted in (Fig. 10). The completion of each phase may trigger one or more events, represented by places *EventOccurredX*. Those places are the interface of the *Timeline* subnet towards the other models. In partic-

ular, each *EventX* place can be connected to a *Precondition* of some *EventOccurrence* subnet.

B. Model Construction

The central idea is to automatically derive the analysis model from the architectural barrier description, and from the WP application. The process for the composition of the overall model consists of several steps, which are described and discussed in the following.

Identification of barriers. The first step consists in the identification of barriers that are relevant for the WP. Barriers are selected based on the *Kind of work*, *Location*, and *Risk*s fields. Based on this, a set of architectural models of relevant barriers, B_i, are retrieved.

Events. For each event present in a barrier model B_i, an instance of the *EventOccurrence* subnet is added to the model. If the occurrence rate (or probability distribution) of the event is known, then a timed activity is used, otherwise *Event* is an immediate activity. In case event recovery is foreseen, an *EventRecovery* subnet is directly connected with the corresponding *EventOccurrence* subnet.

Components. For each component present in a barrier model B_i, an instance of the *GenericComponent* template model is created. Parameters like its possible failure modes, fault occurrence distribution, etc., are derived from the architectural barrier model and the database. In case no fault occurrence information is available for a given component, the corresponding *FaultOccurrence* activity is disabled. If the component is a consumable resource, then the *ConsumableBarrier* subnet is connected to the basic model.

Error propagation. For each failure mode X of a component A, that generates an external fault Y in component B, the corresponding places, *FailureModeX* (in the A subnet) is connected to place *ExternalFaultY* (in the B subnet). In case some delay is specified for the propagation between the two components, an instance of the *EventOccurrence* subnet is added between the two. That is, *FailureModeX* is connected to the *Precondition* interface place, and *ExternalFaultY* to the *EventOccurred* interface place. Propagation delay is given by the firing delay of the *Event* activity.

Propagation of events. Still based on the barrier models, preconditions for events introduced in the previous step are set. In the simple case in which an event has no preconditions, a token is added to the corresponding *Precondition* place, meaning that the event is always enabled. Otherwise, if the event requires a certain combination of conditions to hold, an instance of the *ConditionChecker* subnet is added. The *Condition* place is shared with the *Precondition* place of the *EventOccurrence*. Places *InputX* are instead shared with the events that appear in the condition.

Detection facilities. For each pair *(A,E)*, such that component *A* is marked to be able to detect event *E*, an instance of the *Detector* template is added to the model. The *EventOccurred*, *DetectedOnTime*, and *DetectedLate* places are shared with the ones corresponding to the event to detect, while *Disabled*, and *FailureModeX* are shared with the corresponding ones in the subnet of the component.

Timeline. Timeline models are constructed based on known event that are going to occur during the work. In particular, the occurrence of safety-relevant events are considered, e.g., the isolation of safety mechanisms or the beginning of a different phase of work.

A particular aspect that is modeled using the timeline is the (recent) history of repairs, and the subsequent time that has elapsed until the WP application. In fact, we assumed that for each component we know the last time at which it has been installed, repaired, or replaced. Basically, *the last time at which it could be considered as new*. In case we don't have any updated information on the current state of components, we must refer to such instant of time as their "initial state" when evaluating their future behavior. The oldest repairing time, among those of all the components in the barrier, should then be considered as the starting point of the analysis. With properly created *Timeline* models, in which events occur in correspondence of known repair times, and connecting *EventOcurredX* place with the *DoRepair* place of the corresponding component, this aspect can be represented and evaluated correctly.

Isolation of safety mechanisms. The "Isolation of safety mechanism" field of the WP conveys the important information of which components are disabled during the work. For each event that triggers the isolation (and restore) of a safety mechanism, an instance of the *Disabler* template is added. Places *DisablingStart* and *DisablingEnd* are shared with the places corresponding to the events that trigger the beginning or the end of the disabling phase. These could be, for example, *EventOccurredX* places in the *Timeline* model. Place *Disabled* is connected to the *Disabled* place of *all* the components that are being disabled by this event.

Concurrent work. Work that is being undertaken or is going to be undertaken concurrently to the WP under analysis needs to be taken into account during the evaluation. To this end, the model is extended with elements from WPs that have been already released, but for which work has not been completed yet. The same procedure as in the previous steps is repeated for all concurrent WPs, *reusing* the already introduced template instances for components and events, so that a unique global model is obtained.

Metrics and evaluation. Metrics are specified by the architectural barrier model. Metrics that are evaluated are the union of the metrics specified for all the barriers that are relevant for the WP. These metrics are translated to reward structures, and evaluated at time t_{end}, with t_{end} being the time at which the work specified in the WP application is supposed to end.

VI. Case Study

In this section we show the application of our approach to a real scenario devised with domain experts.

A. The "Gas Leakage" Scenario

One of the hazardous events that might occur during work on a petroleum installation is gas ignition, which may result in very serious damage to people and/or the infrastructure. One of the barriers to avoid such an event is the detection of gas leakage, which may be automated or manual.

We consider a generic area X, in which three gas detectors are operating: A, B, and C. The area can be logically divided in two parts, X1, where A and B are located, and X2, where C is located. Gas detectors A and B are located close to each other, and both are able to detect a gas leakage that occurs in X1, independently from each other. They are also able to detect a gas leakage occurring in X2, but only after gas concentration has become higher, thus only after some delay. Conversely, C is able to detect a gas leakage occurring in X2 in a negligible amount of time; it will also detect a gas leakage occurring in X1, but only after some delay. Gas detectors A, B and C may potentially be of different types and have different failure distributions. Each may fail by missing to detect the gas at all (omission failure) or by detecting the gas, but only after a time delay (late failure).

The presented scenario is a simplification of a real barrier setup, which can include further details, both in the physical and in the cyber dimensions. For example, gas sensors can be decomposed into an initiator (i.e., the physical unit that senses the gas), and a logic solver (i.e., the device interpreting the data, which contains a CPU). Together they are typically considered a single Safety Instrumented Function (SIF), e.g., see [24].

Gas can also be detected manually (i.e., by people on the rig): by smell, portable detectors, or techniques such as smearing liquids on a potential leakage area and looking for bubbles. Gas detectors and manual gas detection form a barrier against the fire ignition caused by gas leakage in area X. We call this barrier "GasDetectionX".

In the scenario we use as example we consider two WP applications for work in area X:

- *WP1*: The work permit under analysis. It is an application for hot work (e.g., welding) to be performed in area X.
- *WP2*: An already released work permit for work within area X. Approved work consists in the replacement of one of the gas sensors. This work would be performed in parallel with the work in WP1.

The evaluation of WP1 proceeds according to the workflow in Fig. 2. The analysis of the *kind of work* (hot work) and of the *location* (Area X) identifies the possible risks and the associated barrier models. In our example, the model for the "GasDetectionX" barrier is retrieved.

A possible representation of the architectural model of this barrier is sketched in Fig. 11. The model includes two events of interest, gas leakage in area X1, and gas leakage in area X2, occurring according to an exponential probability distribution with rate λ_{x1} and λ_{x2}, respectively. The barrier itself is composed of four components: three physical components (gas sensors) and one logical component, representing the manual detection of gas. Sensors failure distributions are exponential, with rate λ_A, λ_B, and λ_C; sensors fail in two possible failure modes: "late" or "omission", with a given probability.

The "Manual Gas Detection" component can detect both the events of interest. The event can be detected on time, detected late, or undetected, with probability values given by parameters *p_manu_ontime*, *p_manu_late*, *p_manu_undetected* respectively. Also gas sensors can detect both events of interest. Their detection probabilities are given by a combination of their working state (healthy, "late" failure mode, or "omission" failure mode) and the event to be detected. For example, if not failed, sensors A and B detect event X1 on time with probability 0.9 and late with probability 0.1, while they will always detect event X2 late, with probability 1.0, since gas has to propagate across the two zones. When failed in the "omission" failure mode, the probability of a gas sensor to detect any event on time, or late is always 0.

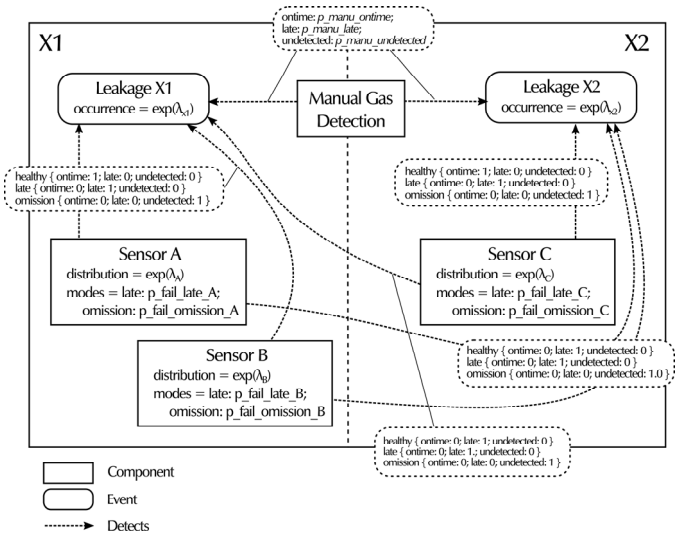

Fig. 11. Architectural model of the GasDetectionX barrier.

Metrics of interest for this barrier relate to the two events of interest: leakage in X1 and in X2. More in detail, for any of the two events E we consider: i) the probability that the event occurs and it is not detected on time, p^E_{late}; ii) the probability that the event occurs and it is not detected at all, p^E_{undet}.

The analysis of concurrent work identifies WP2 as a possible conflicting work, since it operates on components of the same barrier (gas detectors). WP2 is thus included in the analysis model, in order to establish its impact on the risk-related metrics associated with the execution of the work in WP1.

B. Analysis Model Construction

The analysis model is constructed following the procedure in Section V.B. The "Events" step generates 2 instances of the *EventOccurrence* template, one for each of the two leakage events; the "Components" step generates 4 *GenericComponent* instances, one for the manual detection element, and three for the gas sensors. Steps "Error propagation" and "Propagation of events" do not generate in this case additional template instances. Step "Detection facilities" generates 8 *Detector* instances, one for each pair (component,event) such that component is able to detect the event. The "Timeline" step generates 4 *Timeline* instances, one to model the timeline of the WP, and the other three to model the history of the three gas sensors. Finally, the "Isolation of safety mechanisms" and "Current work" steps generate the three *Disabler* instances. As previously explained, all these instances are connected based on their interfaces.

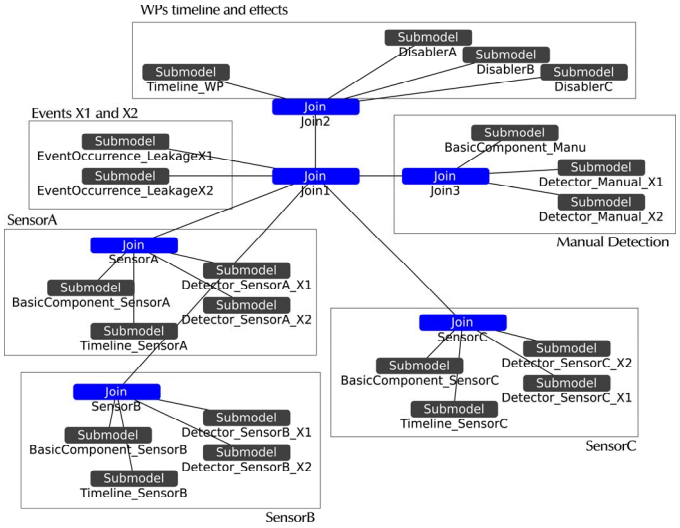

Fig. 12. Generated model for the evaluation of the example scenario.

The obtained model for the scenario of interest consists of a total of 21 instances of templates selected form the library in Section V.A: 2 *EventOccurrence*, 4 *GenericComponent*, 8 *Detector*, 4 *Timeline*, 3 *Disabler* (Fig. 12).

C. Evaluation and Results

The default parameters that we use for the evaluation are reported in Table I. For each of them, we report its default value, the source of information for determining such value, as well as the model template on which it applies. Some parameters, like the probability of manual detection, may be difficult to estimate; however, it should be noted that other state-of-the art analysis approaches suggest to use historical data for similar quantities (e.g., see [6]). Current default values have been devised partially on datasheets, and partially based on our own judgment.

In the evaluation we have performed in this paper we tried to provide insights on the following main questions:

i) How do metrics change if considering only WP1 in isolation, or considering both WP1 and WP2?

ii) How do metrics change if the sensor that is disabled by WP2 is sensor A, B, or C?

iii) What is the impact on metrics of variations of key parameters?

Responding to these questions demonstrate the usefulness of the approach in different system conditions, as those reflect questions that may be raised by decision makers, or to support more long-term planning of work activities on an installation. We assume to have the following constraints to fulfill as indicators of acceptable risk: i) probability of late detection should be below 10^{-4}, and ii) probability of missed detection should be below 10^{-6}. The model has been evaluated with the simulator provided with the Möbius tool [17], with at least 10^7 batches, a relative confidence half-interval of 10%, and a confidence level of 99%.

Results with the nominal parameters from Table I are reported in Fig. 13. Four cases are compared: the case in which only WP1 is considered in isolation, and the cases in which WP2 is included in the evaluation, each time considering a different sensor as the target of disconnection. Based on the above stated constraints, WP1 in isolation could be accepted, although the probability of detecting X2 late is near to the threshold. If considering WP2 as parallel work, WP1 can be accepted only if the sensor that is disconnected is A. In case the sensor that is going to be disconnected is C or B, a warning should be raised instead: in the first case, the probability of detecting X2 late is above the threshold; in the second case both the probabilities of not detecting X1 and X2 are above the threshold. In these cases, the request for WP1 should be rejected.

TABLE I. PARAMETERS OF THE MODEL AND THEIR DEFAULT PARAMETERS

Name	Description	Default	Origin	Template
`lambda_X1`	Occurrence rate of gas leakage in area X1	2.5E-05 h^{-1}	Experts / Historical Data	*EventOccurrence*
`lambda_X2`	Occurrence rate of gas leakage in area X1	2.5E-05 h^{-1}	Experts / Historical Data	*EventOccurrence*
`lambda_A`	Fault occurrence rate of sensor A	9.14E-05 h^{-1}	Datasheets	*BasicComponent*
`p_fail_late_A`	Sensor A – "late" failure mode probability	0.01	Datasheets / Database / Experts	*BasicComponent*
`p_fail_omission_A`	Sensor A – "omission" failure mode probability	0.99	Datasheets / Database / Experts	*BasicComponent*
`last_repair_A`	Time since last repair of sensor A	8000 h	Database	*Timeline*
`lambda_B`	Fault occurrence rate of sensor A	1.14E-06 h^{-1}	Datasheets	*BasicComponent*
`p_fail_late_B`	Sensor B – "late" failure mode probability	0.4	Datasheets / Database / Experts	*BasicComponent*
`p_fail_omission_B`	Sensor B – "omission" failure mode probability	0.6	Datasheets / Database / Experts	*BasicComponent*
`last_repair_B`	Time since last repair of sensor B	3600 h	Database	*Timeline*
`lambda_C`	Fault occurrence rate of sensor A	2.85E-05 h^{-1}	Datasheets	*BasicComponent*
`p_fail_late_C`	Sensor C – "late" failure mode probability	0.6	Datasheets / Database / Experts	*BasicComponent*
`p_fail_omission_C`	Sensor C – "omission" failure mode probability	0.4	Datasheets / Database / Experts	*BasicComponent*
`last_repair_C`	Time since last repair of sensor C	6000 h	Database	*Timeline*
`p_manu_ontime`	Probability of manually detecting gas leakage on time	0.0	Database / Experts	*Detector*
`p_manu_late`	Probability of manually detecting gas leakage late	0.3	Database / Experts	*Detector*
`p_manu_undetected`	Probability of not detecting gas leakage manually	0.7	Database / Experts	*Detector*
`work_duration`	Duration of the work to be performed	12 h	WP1	*Timeline*
`isolation_start`	Time after beginning of work in WP1 that the sensors would be dissconnected	4 h	WP1 & WP2	*Timeline*
`isolation_duration`	Duration of the isolation period	6 h	WP1 & WP2	*Timeline*

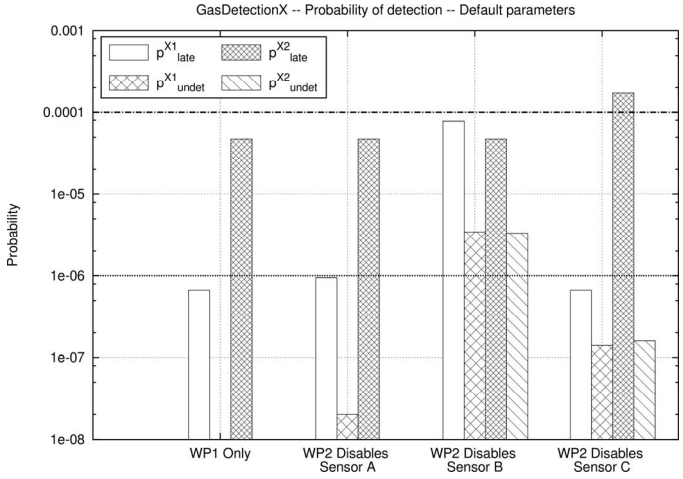

Fig. 13. Probability of not detecting gas leakage in the four cases – nominal parameters.

In the following figures we evaluate the impact of varying the fault occurrence rate of one of the sensors, in particular Sensor C, on the metrics of interest for the GasDetectionX barrier. Results for the case in which only WP1 is considered are reported in Fig. 14. In general, a higher fault occurrence rate for C increases the probability that one of the two events of interest is not detected or it is detected late. The effect is much more evident on the metrics for X2, since sensor C is the only one that is installed in area X2. However, almost for all the considered range all the constraints are fulfilled; only for λ_C = 2.85E-4 or higher, the probability of detecting X2 late is slightly above the threshold.

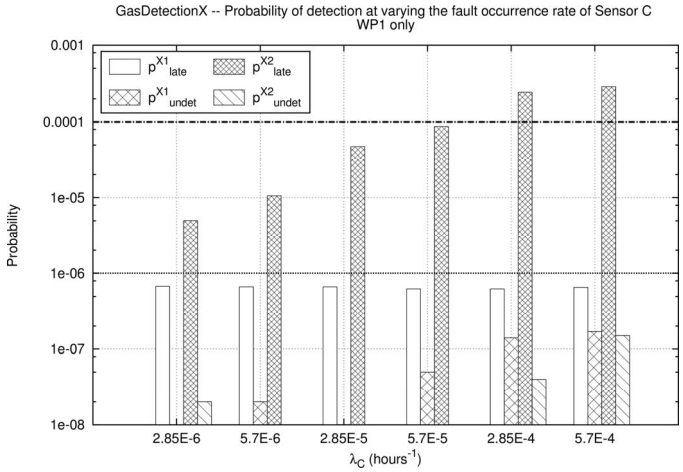

Fig. 14. Probability of not detecting gas leakage at varying the fault occurrence rate of Sensor C, considering WP1 only.

The case in which WP2 is executed in parallel, leading to the disconnection of sensor B is shown in Fig. 15. In this case, the impact of varying λ_C is much greater, and it impacts not only the capability of detecting a leakage in X2, but also the capability of detecting a leakage in X1. This is exactly due to the fact that B, which is one of the two sensors located in area X1 is going to be disabled by the parallel execution of WP2. If taking into account the effect of WP2, any value for λ_C that is equal or higher than 2.85E-5 violates the constraints, i.e., an order of magnitude lower is required with respect to considering WP1 in isolation. However, with an even lower λ_C, WP1 could still be accepted even in parallel to WP2.

Besides helping decision makers to accept or reject a work permit, this information can also be used to improve the barriers system. If work that needs to disconnect sensor B needs to be executed frequently, one of the countermeasures could be the replacement of sensor C with one of greater quality (i.e., lower fault occurrence rate).

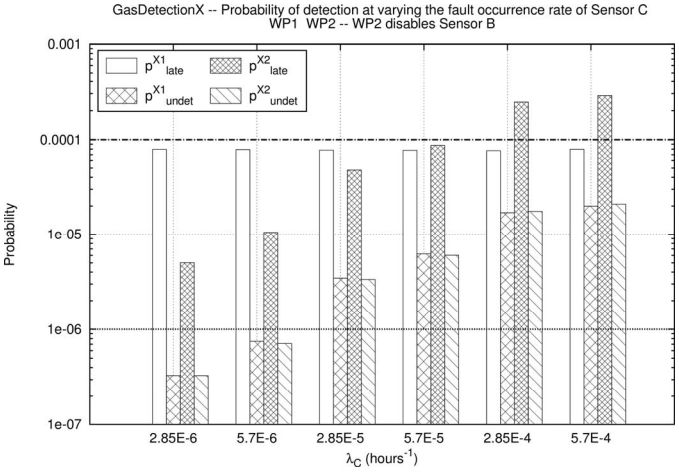

Fig. 15. Probability of not detecting gas leakage at varying the fault occurrence rate of Sensor C, considering both WPs, with WP2 disabling Sensor B.

VII. Automation and Integration

The analysis that we presented in this paper has been executed manually, i.e., model templates for the gas leakage scenario have been instantiated and connected manually, and then analyzed using the Möbius [17] tool. Automation can be performed with current technology, and is being undertaken as part of a research project. Our plans for the implementation are discussed in the following.

One of the aspects that we left as an implementation detail in this paper is how barriers models are actually designed by the domain experts, both because it can depend on the preferences of the individual company, and because there are no established languages yet. For our realization, we will use a custom UML profile, with specific extensions to model the information described in Section IV.B. This resulted to be the most convenient solution for a prototype, since in this way we can reuse a lot of concepts and tools that are popular in the UML domain. For example, we can use the popular Papyrus [20] diagram editor, without the need to create a specific editor or file format. At the same time, the domain-expert modeler will not need to learn a completely new language. The library of barriers architectural models is thus a collection of UML files. It should be noted however that UML is not a prerequisite of our approach: any custom DSL could be used instead.

Concerning analysis models we chose, for practical reasons, to store the library of SAN template models as a collection of XML files, since it is the format in which the Möbius tool stores the definition of SAN models. The selection of needed templates, generation of instances, and their composi-

tion is a model-transformation step; as such, it is performed by a combination of the ATL [18] and XSLT [19] languages, which are transformation languages operating at model-to-model and text-to-text levels, respectively. The final analysis of the composed barrier models is performed by solvers included in the Möbius framework.

VIII. Conclusion

In this paper we have proposed an approach to support decision makers in accepting (releasing) or rejecting work permits in the petroleum domain. The support is provided by using state-based stochastic methods. The model of the scenario to be analyzed is automatically created combining together multiple instances of a basic template models library, based on the safety barriers of interest, the WP to be analyzed, and possible concurrent WPs that have already been released. The analysis of the model provides the decision makers with metrics that can guide them in processing the WP application objectively. We applied the approach to a simple but representative example of a gas leakage scenario, focusing on the impact of two concurrent WP applications.

The work in this paper represents a first step in addressing the WP release problem using stochastic evaluation. To completely address this problem, several challenges still need to be undertaken, and they are part of our next planned steps. Evaluating the models by simulation may require significant time, especially because the focus is on events that are supposed to be rare. We plan to investigate this aspect more in depth, and possibly resort to advanced analytical solution methods when possible. A longer-term activity consists in understanding if further information can be exploited for building the analysis model, and possibly extend the SAN model templates library. Addressing this challenge is needed but difficult, as it requires us to develop a deep knowledge of the information stored by different companies, for different kinds of installations.

Acknowledgment

This work has been partially supported by the ARTEMIS-JU CONCERTO project (n.333053), and by CECRIS, "CErtification of CRItical Systems", FP7–Marie Curie (IAPP) number 324334.

References

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE Transactions on Dependable and Secure Computing, pp. 11-33, January-March, 2004.

[2] G. Ciardo, R. German, C. Lindemann, "A characterization of the stochastic process underlying a stochastic Petri net" IEEE Trans. on Software Engineering, 7, 1994, 20, 506-515.

[3] Norwegian Oil and Gas, 088 – Recommended Guidelines for a Common Model for Work Permits (WP), rev. 5, 22/06/2015.

[4] T. Aven, S. Sklet and J.E. Vinnem, "Barrier and operational risk analysis of hydrocarbon releases (BORA-Release)". Journal of Hazardous Materials, Vol. 137, Issue 2, pages 681-708, September 2006.

[5] W. Røed, A. Mosleh, J.E. Vinnem and T. Aven, "On the use of the hybrid causal logic method in offshore risk analysis." Reliability Engineering and System Safety, Vol. 94, Issue 2, pp. 445-455, February 2009.

[6] J.-E. Vinnem, "Offshore Risk Assessment vol 2.: Principles, Modelling and Applications of QRA Studies", Springer Series in Reliability Engineering, 2014.

[7] D.M. Nicol, W.H. Sanders, K.S. Trivedi, "Model-based evaluation: from dependability to security", IEEE Transactions on Dependable and Secure Computing, 1, 2004, 1, 48-65.

[8] S. Bernardi, J. Merseguer, D.C. Petriu. "Dependability modeling and analysis of software systems specified with UML". ACM Comput. Surv. 45, 1, Article 2, December 2012.

[9] B. Gallina, E. Sefer, A. Refsdal, "Towards Safety Risk Assessment of Socio-Technical Systems via Failure Logic Analysis," in Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on pp.287-292, 3-6 Nov. 2014.

[10] A. Refsdal, Ø. Rideng, B. Solhaug, and K. Stølen. Divide and Conquer-Towards a Notion of Risk Model Encapsulation. In Engineering Secure Future Internet Services, Vol. 8431, London: Springer, 2014.

[11] W. Sanders and J. Meyer. "Stochastic activity networks: formal definitions and concepts". In: Lectures on formal methods and performance analysis. Vol. 2090. LNCS. Springer, 2002, pp. 315–343.

[12] E. W. Dijkstra. "On the role of scientific thought". In:Selected Writings on Computing: A Personal Perspective. Ed. by E. W. Dijkstra. Springer, 1982, pp. 60–66.

[13] K. Kanoun and M. Ortalo-Borrel. "Fault-tolerant system dependability-explicit modeling of hardware and software component-interactions". In: IEEE Transactions on Reliability 49.4 (2000), pp. 363–376.

[14] M. Rabah and K. Kanoun. "Performability evaluation of multipurpose multiprocessor systems: the "separation of concerns" approach". In: IEEE Transactions on Computers 52.2 (2003), pp. 223–236.

[15] N. Veeraragavan, L. Montecchi, N. Nostro, R. Vitenberg, H. Meling, and A. Bondavalli. "Modeling QoE in Dependable Tele-immersive Applications: A Case Study of World Opera". In: IEEE Transactions on Parallel and Distributed Systems (2016). To appear.

[16] L. Montecchi, P. Lollini, and A. Bondavalli. "A DSL-Supported Workflow for the Automated Assembly of Large Stochastic Models". In: Proceedings of the 10th European Dependable Computing Conference (EDCC'14). Newcastle upon Tyne, UK, May 13-16, 2014.

[17] T. Courtney, S. Gaonkar, K. Keefe, E. Rozier, W. Sanders, "Möbius 2.3: An extensible tool for dependability, security, and performance evaluation of large and complex system models", 39th IEEE/IFIP International Conference on Dependable Systems Networks (DSN'09). , 2009, 353-358

[18] Atlas Transformation Language (ATL). http://www.eclipse.org/atl/ (Accessed: 07/12/2015).

[19] XSL Transformations (XSLT) Version 2.0, W3C Recommendation 23 January 2007.

[20] S. Gérard, et al., "Papyrus: A UML2 Tool for Domain-Specific Language Modeling Model-Based Engineering of Embedded Real-Time Systems", Springer Berlin Heidelberg, 2011, 6100, 361-368.

[21] Object Management Group. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.5. OMG Document {formal/15-03-01}. June 2015.

[22] L. Montecchi, N. Nostro, A. Ceccarelli, G. Vella, A. Caruso, A. Bondavalli, "Model-based Evaluation of Scalability and Security Tradeoffs: a Case Study on a Multi-Service Platform," Electronic Notes in Theoretical Computer Science, vol. 310, pp. 113-133, January 2015.

[23] A. Bondavalli, O. Hamouda, M. Kaâniche, P. Lollini, I. Majzik, and Hans-Peter Schwefel. "The HIDENETS Holistic Approach for the Analysis of Large Critical Mobile Systems." In IEEE Transactions on Mobile Computing, vol. 10, Issue 6, pp. 783–796, June, 2011.

[24] The Norwegian Oil Industry Association, "Application of IEC 61508 and IEC 61511 in the Norwegian Petroleum Industry" No. 070, Revision no. 02, October 2004.

[25] M. Kaâniche, P. Lollini, A. Bondavalli, and L. Kanoun. "Modeling the Resilience of Large and Evolving Systems." In International Journal of Performability Engineering, Volume 4, Number 2, pp. 153-168, 2008.