A Safety Analysis Method for Multi-Agent Systems

Ching Louis Liu, Edmund Kazmierczak, Tim Miller

Abstract—Safety analysis for multi-agent systems is complicated by the, potentially nonlinear, interactions between agents. This paper proposes a method for analyzing the safety of multi-agent systems by explicitly focusing on interactions and the accident data of systems that are similar in structure and function to the system being analyzed. The method creates a Bayesian network using the accident data from similar systems. A feature of our method is that the events in accident data are labeled with HAZOP guide words. Our method uses an Ontology to abstract away from the details of a multi-agent implementation. Using the ontology, our methods then constructs an "Interaction Map," a graphical representation of the patterns of interactions between agents and other artifacts. Interaction maps combined with statistical data from accidents and the HAZOP classifications of events can be converted into a Bayesian Network. Bayesian networks allow designers to explore "*what it*" scenarios and make design trade-offs that maintain safety. We show how to use the Bayesian networks, and the interaction maps to improve multi-agent system designs.

Keywords—Multi-agent system, safety analysis, safety model.

I. Introduction

As the popularity of multi-agent systems increases so too does the number of agents interacting with humans in a variety of different industrial settings. Many of these, such as industrial robots, air traffic control and nuclear control systems are safety critical. A safety critical system is a system that can cause unintended harm, regardless of whether or not there is a system failure. That is to say that a safety-critical system can cause unintended harm just by operating normally as well as in the case of a system failure. Accidents are events that result in the type of harm that we wish to avoid. The analysis of such events forms part of the accident investigations where causes are associated with each contributory event. The events, causes, circumstances and timelines associated with accidents constitute the accident data.

Software agents cannot harm or physically injure people without some form of physical interaction. Software by itself does not do this kind of harm; however, once it is controlling a physical machine or some other device then the story is very different. A physical agent when interacting with humans can cause this kind of harm and accidents involving physical agents are well documented [12].

To reduce the likelihood of accidents, we propose a design theory that quantifies the accident likelihood using accident data from similar accidents in the past and a system ontology that captures the taxonomy of components and processes making up the type of system being analyzed. In more traditional forms of safety analysis, frameworks such as HAZOP [13] are used to identify hazards and assign a level of risk to each hazard. Hazards are those situations or states of a system immediately preceding an accident, for example, a hazard is a lighted match in a room where a flammable gas is leaking. The hazard becomes an *accident* if a certain event, or events, occurs while in the hazardous state, but that event need not occur and then we are left with an *incident* and not an *accident*. In the case of a match in a room with leaking flammable gas, then an accident occurs if the flame from the match comes into contact with the gas, but if the lighted match is extinguished before coming into contact with the flammable gas then the accident has been averted, and we are left with an incident - a lighted match in a room with leaking flammable gas - but no harm.

Our assumption is that most accidents are not new and that the data from accidents can be systemised and reused. Our method attempts to make use of this assumption by using the *PhySys* system ontology [10] to structure the safety analysis for a specific system, generalize the accident data for that system. A key feature of multi-agent systems [1] is the interaction between agents to achieve system-wide goals. However, from a safety perspective the interaction between agents a major concern for safety. Our method addresses the issue of interactions between agents directly through *Interaction Maps* and *Bayesian* networks and uses this analysis to generate the safety requirements.

II. Related Work

Generally, agent-oriented software engineering methodologies consider safety concerns as non-functional requirements (NFR) [1]-[3].

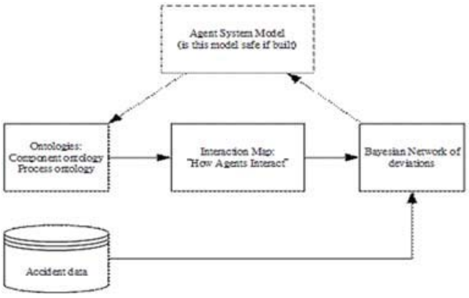


Fig. 1 Proposed MAS safety analysis

Ching Louis Liu is with the Department of Computing and Information Systems, The University of Melbourne, Parkville Australia 3010 (e-mail: louismail@gmail.com).

Edmund Kazmierczak and Tim Miller are with the Department of Computing and Information Systems, The University of Melbourne, Parkville Australia 3010 (e-mail: edmundak@unimelb.edu.au, tmiller@unimelb.edu.au).

A number of papers suggest different architectural approaches to constructing systems that meet and handle these NFRs. In [4] a method for identifying all of the system modes (e.g. start up, shut down, maintenance) for an embedded system in a systematic way is proposed. They then propose a method for capturing safety requirements by examining the different system modes. In [5] a multi-agent framework is proposed that translates safety concerns into actions once a hazard is encountered.

Failure mode and effects analysis (FMEA) is a hazard analysis method that is based on identifying how component failures lead to system hazards. A number of multi-agent systems analysis methods have been proposed based on FMEA. In [6] a multi-agent structure to manage hazards based on FMEA analysis is proposed, while [7] proposes a multi-agent system using a language based on FMEA to self-management system failures. Although we may observe some similarity between the UML-like expressions in [7] and our own *Interaction Maps* (defined below), the language in [7] tries to capture a much wider range of system behaviors while interaction maps deliberately limit themselves to focus on behaviors influencing safety. Another approach is that taken in [9] which uses temporal logic to formalize agent behaviors, and to explore behaviors that may lead to hazards.

In this paper, we assume the ROADMAP methodology for developing multi-agent systems from [8]. The major difference between our approach and previous work is that none of the methods listed above deal directly with the problem of generating and assessing safety requirements, while our method provides a clear and quantitative way of constructing and assessing safety constraints for MAS. Further, while our method is based around ROADMAP, it can be easily adapted to all the methodologies and frameworks for MAS engineering such as [2].

III. A Method for Safety Analysis in Multi-Agent Systems

The proposed method for safety analysis for MAS uses accident data from previous accidents together with appropriate systems ontology to quantitatively assess new agent-oriented system designs for hazards. The types of accidents used in the analysis are related to the type of system begin developed; for example, although technology has advanced dramatically for cars over the years, the types of accidents related to cars has remained remarkably unchanged throughout this time: front impact, side impact, rear impact, rollover and fire being the most common. Changes in automotive technology may have reduced the incidence or severity of some types of accidents, or the types of faults may have changed alongside these technology changes, but ultimately the types of automotive accidents still involve collisions and fires.

The role of the ontology of the system is to provide a clear description of how different components work together and how they interact with each other. Consequently, the system ontology forms one of the key tools for deducing how system behavior and interaction leads to accidents. We propose a safety analysis method for Multi-agent systems that first uses the system ontology combined with previous accident reports to create a safety knowledge base. Then we use this safety knowledge base is used to provide design guidance and safety analysis for the different agents in the system. With the information from accident reports as well as any regulations and legal requirements, our approach aims to provide statistical constraints for agent designs and the ability for agents to collect data to improve the knowledge base.

IV. The System Ontology

The system ontology specifies how things interact in the given system. Although there are a number of different ontologies proposed in the field, all of them can generally be separated into two interdependent sub-ontologies:

1. A *Component Ontology*; and
2. A *Process Ontology*.

The *Component Ontology* defines the main types of components, or building blocks, of the system and their properties as well as any relationships between these components. The *Process Ontology* defines how different components interact to perform the system functions on more temporal dimensions.

In this paper, we adapt a simplified *PhySys* [10] ontology for the purpose of exposition, but our method can use other ontologies as well. The reason an ontology is required is that accident reports collected in the field about a type of system come from different machines, models, environments, and companies. This results in numerous different details between different accident reports. To use this data effectively to form accident knowledge we need to generalize it to a common format, and this is where the component ontology comes in. Now observe that, at a certain level of abstraction, in any given process, although it is carried out with different components in different agents, the fundamental process remains the same if the kind of system is the same. This can be represented by a *process ontology*. We will use process ontologies to represent the flow of energy in a system. By analyzing the process ontology and the steps in the abstract process and tying these to the corresponding events in accident reports, we can unify all accident reports for a particular kind of system into one common representation. As an example of this latter idea consider two different cars. One car uses petrol, while the other uses a hybrid power source. Although they use different power sources both of them are governed by the same macro process ontology - driving a car in this case - in which case collision events for both types of care can be classified into the same ontological category at a certain level of generality.

V. Using HAZOP to Generalise Deviations from Intent

With the generalized processes in the *PhysSys* ontology we also need a generalized notion of a deviation from intent, because, as many common hazard analysis methods suggest, it is the deviations from intent that lead to hazards. The system of deviations from intent used in our work comes from HAZOP's. Hazard and Operability Studies (HAZOP) is a technique for identifying the hazards of a system by examining deviations from intent [13]. It is a well-established technique for Preliminary Hazard Analysis (PHA) whereby a set of *guide-words* is used to explore the behavior of a system and the causes and consequences of the deviations. The guide-words are used in traditional hazard analysis to prompt the analyst to explore a particular deviation.

In this paper, we use the guide-words differently. Instead of prompting an analyst to explore a behavior, the guide-words will be used to define the *state* of a component in the system. The specific guide-words used to come from [13] and are summarized in Table I. Another way of classifying deviations is the *deviation type* as follows:

1. Temporal (Early, Late),
2. Quantitative (More, Less)
3. Sequential (Before, after)
4. Omission (None, Part of)
5. Commission (Other than, as well as)

The deviation areas can also form the states of a component but at a higher level than an individual deviation. As an example, it may not matter whether an action occurs *Early* or *Late,* just that it is out of the temporal order. Further, individual deviations can be combined to record the fact that more than one deviation occurred in an event.

TABLE I
HAZOP KEYWORD

Keyword	Meaning
None	None exist
More	More than normal
Less	Less than normal
Early	Earlier than normal
Late	Later than normal
Normal	Normal State
As well as	There are more than normal
Other than	Instead of normal behavior, other behavior occurred

VI. Classifying Accidents

Accidents are defined in the safety literature as unintended events that result in harm or loss [12], [14]. A key assumption in this work is that accidents are not new. Most, if not all, *types* of accidents have occurred before, and therefore knowledge bases of accidents can be built up and used in safety analysis. First, however, we need to understand what to classify as an accident in an accident report. Not all of the information contained in accident reports will pertain to the accident. An accident report could contain unrelated information that bears no causal relationship to the accident, or accident reports may omit some information pertaining to the cause of the accident. As a result, we need to identify what part in which states can be classified as an accident in accident report data, and the causal relationship between different nodes/parts that resulted in an accident.

Accident reports analyze accidents as a chain of events. Such data can be separated into two distinct parts, core accident factors and an influence path. The factors involved in the classification of an event, like an accident, are known as the *Core Accident Factors*, while the I*nfluence Path* is the sequence of events leading to the similar situation in the past. For example, one particular type of car accident is a car head on collision. For a head on collision to occur, the car is required to be moving, and an obstacle or another car is required to be in its path. The state of a component of the car in question, such as its brake, may well be in a deviant state according to our HAZOP table. However, there are many different chains of events that can lead to the same condition. In this example, the deviations from intent in the states of the car and the other objects are core accident factors, and event chains that lead to such an accident are the influence path.

To separate core accident factors from an influence path is a learning problem in itself. We solve this learning problem by adopting an *Augmented Naive Bayes* [11] learning algorithm augmented with our system ontology. The Augmented Naive Bayes learning algorithm allows for the relaxing of constraints on conditional independence while still being able to learn the architecture of different nodes. It has better accuracy compared to the Naive Bayes learning algorithm although requires a longer computation time. The Augmented Naive Bayes produces the degree of correlation between different factors and the accident, as well as the structure of different factors related to each other. System Ontology defines the point of interactions in the system.

A critical assumption is that an accident can only occur if the interaction between humans, or other objects that can suffer harm or loss, and the system is in a deviant state. We can easily identify these nodes as being among the core accident factors, while other information in accident reports are a part of the influence path.

A typical analysis from an accident report is a causal event chain [15], and this event chain can be represented in many different ways. Core factors and the influence path capture this causal event chain in the augmented naive Bayes framework that we use, but this causal chain learned from a collection of accident reports will need to be checked against the system and process ontology to verify that it is possible, and that the inference engine has not learned an improper connection. Further, the system ontology to help can be used to guide the learning process by interpolating steps known to be part of the system's behavior (from the process part of the ontology) when they are missing from an accident report.

VII. Using the Accident Model

The accident model can be used in one of the folioing ways. For an existing system, we can use any available accident data to determine probability distribution functions for accidents. From the probability function, the likelihoods of accidents can be predicted and the estimated costs of accidents calculated.

For new systems, we don't have the data for all of the states and functions of the system. In this case, we would ideally like to determine the minimal assumptions necessary for meeting legal or design requirements. Notice from Table I that we have stated for all of the deviations from intended *normal* behavior. What we do not have in Table I is the *normal* state, and as a consequence, we need to add it to our set of component states.

Our problem, however, is that accident reports contain a great deal of information on how deviations propagate and the likelihood of propagation, but contain no information on the likelihood of staying in a normal state! To determine the probability of a "*normal to normal*" transition, we make the weakest possible assumption that meets the constraints of the design and legislation. The designer can then quantitatively optimize the design such that it is better than these minimal estimates.

In this paper, we will deal more with the second scenario, which is to imitate an operational statistical model with safety requirements and calculate its safety constraints. To provide enough data for a new system, we have the following assumption: accidents only happen when there are deviations from intention. We will further assume that the intentions have been adequately captured in the agent specification. In general, the environment in which the system operates is difficult to quantify or model accurately as a separate component, but our theory takes the environment into account through the probability distributions of states. In our model, the different environment states are captured within the probability distributions in the nodes. For example, it is not possible to have an aviation disaster if all parts of the plane and the environment the plane is operating in are in normal states.

Given the assumptions above and the idea that normal to normal state transitions are defined according to meet external safety constraints then the ensuring that the design of the new system is within a given quantitative safety level becomes a Maximum a Posteriori estimation (MAP) problem [11]. It is used in our case to estimate the accident probability of the new system given the data in the accident reports.

VIII. Interaction Maps

An Interaction Map is a graphical representation of how agents interact to accomplish a task. It is the first step in the safety analysis and is composed of three different types of components:

1. Actions
2. Resources
3. Agents

Actions define the external behaviors of that the system can perform. *Resources* are materials or signals generated or requested by actions. *Agents* are groups of actions and resources combined together. Although different agent methodologies propose different ways to identify system functionalities and groupings of actions, all of them can be represented by an *Interaction Map*. Interaction Maps act as the lowest common denominator to represent system functionalities and the manner in which different agents interact together.

In this paper, we propose a simple way to construct an interaction map that can easily be adapted to other methodologies. The *Interaction Map* is based on the process ontology and accident report descriptions. The process ontology comprises the backbone of how different components interact together to perform a task. Accident reports outline the parts of the process ontology that require more attention as they have been involved in accidents before. An Interaction Map is a graph with nodes being the constructed by the procedure below:

1. Write down all the functions in the process ontology; these become the *agent actions*;
2. Write down the resources required for the system;
3. Identify which actions act on resources and link these with an edge;
4. Group similar or related actions together to form an *Agent*
5. Check if the actions can be broken down into finer actions for a better representation of the accident
6. Repeat step 4 and 5 until interaction map covers all the accident cases and agents are grouped in a logical manner

Agents are a combination of actions and resources, the connection between different actions will be though resources.

IX. Converting the Interaction Map into A Bayesian Network

An Interaction Map is a graphical representation of the process ontology. Each node in the interaction map is either an action or a resource that is consumed by, or produced by, actions. This shows a relationship between different actions and resources. We can represent the relationship in a statistical manner. We choose Bayesian Network due to the fact that:

1. Bayesian Networks (BN) have a similar graphical representation of interaction maps; and
2. Each node in a Bayesian Network contains different states, same as our interaction map.

However, Bayesian Networks are an acyclic graphical model, which means that cycles, or loops, are not allowed in the network while cycles are possible in interaction maps. A *Dynamic Bayesian Network* can be used to resolve the cycles in the Bayesian Network that arise from the cycles in an interaction map. In essence, each cycle in the interaction map is '*unwound*' in the Dynamic Bayesian network.

The probability of a state occurring at a node in the Bayesian Network can be calculated by counting the occurrence of the state in accident reports. For example, if node ***Press*** is in the state ***More*** three times in accident reports, and the total number of occurrences of node ***Press*** is five times in total, then the probability of state ***More*** occurring is 3/5 in node ***Press***. As we mentioned before if the state of a given node is missing in accident reports, then we assume it is in the ***Normal*** state in the initial iteration.

The statistical model built so far is "negative" as it represents a statistical accident model; it gives the most probable explanation of what did happen when an accident occurs. To design an agent system, it will be more useful if we provide a "positive" model to calculate the limitations of actions and resources of agents. This can be achieved by defining the minimum safety level of the system, then combining an accident model with normal data to form a "positive" statistical model.

X. The Mathematical Argument

Our method uses accident reports complemented by the probability of a ***Normal*** state occurring. Doing this allows us to calculate accident probabilities relative to the normal use of the system and to use these probability distributions as a guide to safe system design. The aim is to ensure that the probability of no accident occurring is greater than or equal to the real world situation, so designs based on our method will always be conservatively safe.

The probability of an accident occurring from given accident data and complemented by estimates of the probability of a ***Normal*** state is given in (1).

$$P(A) = \frac{R}{(R+N)} \tag{1}$$

where: R: Number of accident reports (Number of recorded accidents); N: Normal States.

As a result, the accident rate is the number of accidents cases as a proportion of the total number of cases, and we can adjust the accident rate by adjusting the number of normal states of the system. The real accident rate is not known as not all accidents are reported and recorded, and not all reports are combined into one single source for reference. However, our accident rate model can be adjusted to meet the minimum desired or legal requirements and consequently a design based on our model can be made safer in real life situations.

In the sequel, we give an example to illustrate the idea. For the simple case of two nodes (A and B) suppose each node has three states (M, L, N).

From accident reports, we see that only when node B is in state M will the result be an accident, and only two types of accidents are possible. Table II shows the counting of state M in two types of accident.

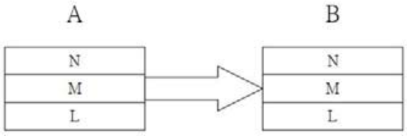


Fig. 2 Simple example of accident probability calculation

TABLE II
ACCIDENT REPORT DATA FOR SIMPLE EXAMPLE

Node	State	Count
Accident Type I		
A	M	2
B	M	2
Accident Type II		
Node	State	Count
A	L	8
B	M	8

We can see that there are 10 accident cases, 2 of them are due to node A in state M while 8 of them are due to node A in state L. As a result, we know the contribution to an accident of node A is:

- If the minimum legal requirement is 1 in 100, then to make it safe we need additional 990 normal cases.

The safety design goal will become:

- For node B: State M cannot occur more than 1/100 = 0.01
- For node A: State M cannot occur more than 1/500 = 0.002; State L cannot occur more than 8/1000 = 0.008

From accident reports we can see that node A will affect node B. As a result, we can assume that by limiting node A, we achieve the desired safety objection.

XI. An Ontology for the Industrial Press

To demonstrate the methodology in practice, we give an example of the *Industrial Press*, first introduced in [14]. Our *Industrial Press* model consists of seven internal components and three external components, where each of these components may, in turn, contain smaller components. The components are connected as in Fig. 3.

The flow of energy begins from electrical energy, then converted into Harmonic motion by motors inside the press to slide and die, then energy is transformed to deform the material physically and later more energy is spent to extract the final product from the press.

XII. An Agent-Based Model and Its Interaction Map

With the *PhysSys* ontology combining both a taxonomy of components and a Process ontology, we can design the press with the following diagram. Suppose we have two agents, one is Worker, and the other is Press. There are four actions the worker agent can perform, namely:

1. Input material
2. Activate Press
3. Received Pressed Signal
4. Extract Product

Those actions are connected to Press agent's action via 4 different materials, they are:

1. Raw material
2. Press Signal
3. Finish press signal
4. Product

An internal material in both agents is the Safety Guard that prevents unwanted interactions between the two agents at different times. The Press agent has two actions:

1. Accept Material
2. Press Action

These actions and materials are closely related to the component and process ontology for press systems. Further, different materials and actions involving agents are related to different components and process steps. The interaction map specifies exactly how the various actions, materials, and agents interact on a task.

In our case, the interaction begins with a Worker agent that inputs material into the Press agent who accepts it when it is ready. Then the worker will activate the press by sending a ‘press’ signal to the Press agent. Once the Press agent has received the ‘press’ signal, it will activate the stamping action. After that, a ‘finish press’ signal will be sent to the worker agent to indicate the press action is completed. Then the worker agent will extract the product from the press agent, and the pressing cycle starts again.

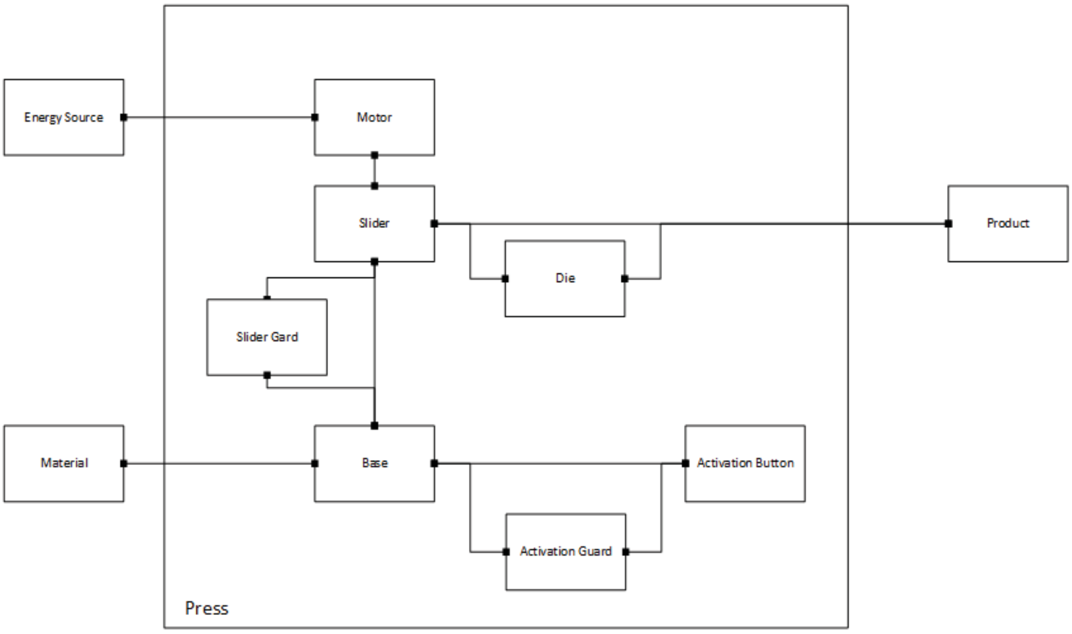


Fig. 3 Press Component Ontology

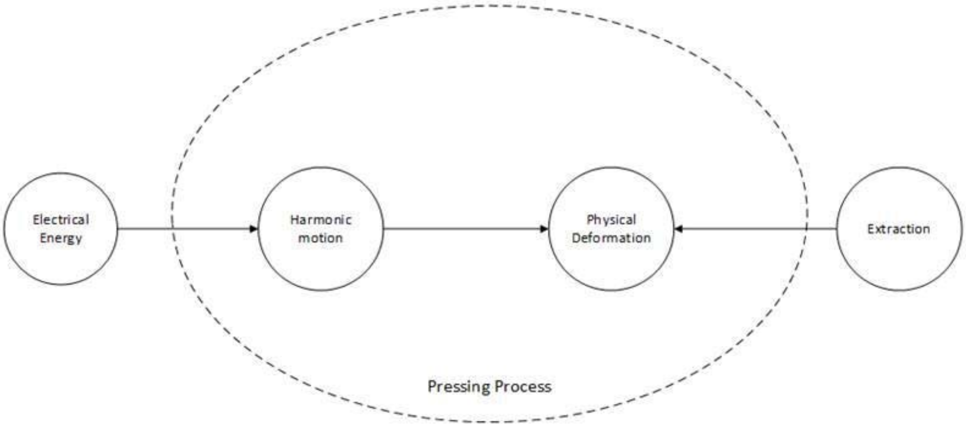


Fig. 4 Press Process Ontology

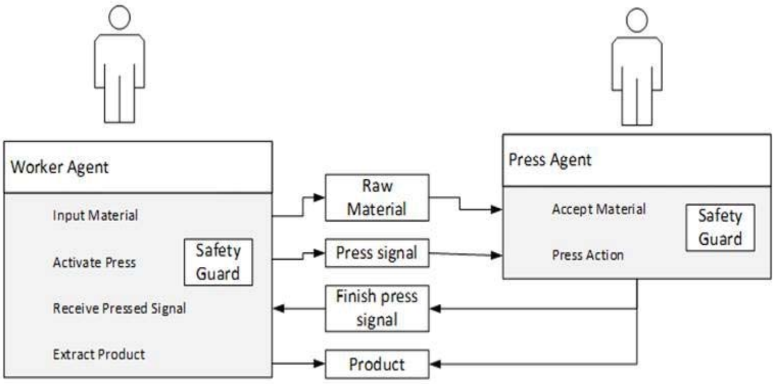


Fig. 5 Interaction Map of Press

XIII. The Bayesian Network for the Press

With the interaction map defined above, we can construct the following Bayesian Network.

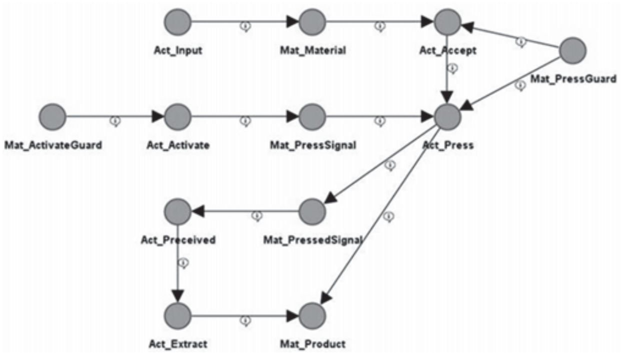


Fig. 6 Bayesian Network

Relationships between different nodes are governed by the interaction map stemming from the system ontology. An immediate result of the use of the system ontology is that we can predetermine the relationship between different nodes. Secondly, the probability between different nodes can be learned from accident reports using the counting technique that we noted above. We translate accident reports based on Interaction Maps and assign different states to different nodes in the system. Consider, as an example, the following case: *Employee #1 was operating a 25-ton L & J full-revolution mechanical power press. The employee was seated and stamping parts when a part became stuck in the point of operation. Employee #1 was reaching in with his right hand when he stood up and apparently stepped on the foot pedal, activating the press. His index and middle fingers were amputated to the first joint, and his ring finger was fractured.*

The use-case is converted into the following where we only show the deviant states.

Worker agent: *Activate = Early;*
Extract = More (More than usual)
Press agent: *Press = Early*

This set of states will then feed into our Bayesian Network model to propagate the probabilities through the network. Classification of accidents is done by assigning an additional node to the Bayesian network with the accident outcomes as indicated by the accident reports. As each accident report is a record of an accident, logically we can express accidents in the following formula:

$$(C_{1a} \wedge C_{1b} \wedge \dots) \vee (C_{2a} \wedge C_{2b} \wedge \dots) \vee \dots \qquad (2)$$

where: C_{1a} is accident condition *A* of accident report *1*; C_{1b} is accident condition *B* of accident report *1*; C_{2a} is accident condition *A* of accident report *2*. This is the logical summation of all the conditions of all the reports, and can be simplified due to the fact that lots of accident reports contain the same accident condition. Core accident factors can then be identified by the combine with process and component ontology as those factors can only occur at interactions involving humans.

With our assumption of not all accident report data are correct, to verify the data and reduce learning noise, we can exam accident report data quality by the following:

1. Test if there exists common accident conditions by separate data sets into learning data and test data, we can build the model from learning data and measure against testing data set.
2. Determine how different conditions in the accident report contribute to the accident, by measuring the accident contribution of each condition.

In this case, we have a total of 108 accident cases; we use 80% as learning data with the remainder of testing data. Augmented Naive Bayes learning algorithm allows relaxing constraints on conditional independence while being able to learn the architecture of different nodes. It has better accuracy compared to Naive Bayes learning algorithm although require more computation time. We use Augmented Naive Bayes method to perform an unsupervised learning on the accident data and obtained following: The model learned with sample data is closely matched with Test data with 95.24% accuracy, which indicates that our model captures different accident modes in the accident report.

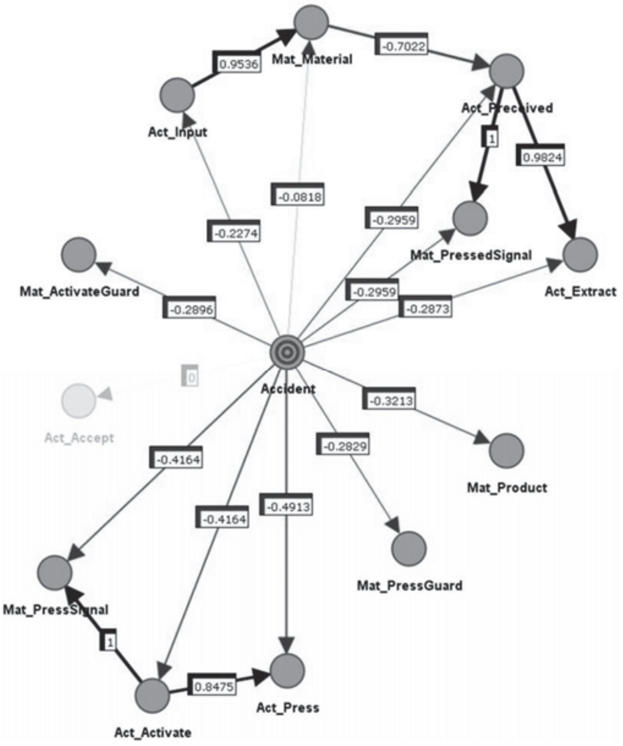


Fig. 7 Augmented Naive Bayes of Accident

The node in the center is accident node, and all the nodes surrounding it are nodes in the system. The arrow indicates the parent-child relationship. The thickness of the arrow indicates the correlation between nodes. Thick arrow indicates the relationship is positive; while thin indicate the relationship is the reverse. The number of the link indicates the weight of correlation.

From Fig. 8, we can see Agent action Active (*Act_Active*) is the parent of Press Signal material (*Mat_PressSignal*) and Press agent Press action (*Act_Press*). This makes sense as the press signal, and press action will depend on activating press action.

The relationship between them is positive as once you press the activation button a signal will be sent to the press machine and a press action will be performed. This also explains why the relationship between different nodes and accident nodes is negative as accidents will only happen when a component deviates from the original. The strength of the correlation is indicated by the label on the arrow. We can see Activate action is 100% correlated to Press signal material, so every time when press button is pushed a signal is sent to the press, while the press will not perform the Press action correctly every time (as the correlation is 0.8475).

We can reduce the noise of data by setting a threshold on the correlation between accidents and nodes in the system. So if the correlation between a given node and accident is low, then we argue this node is less related to accidents, and hence require less monitoring compared to others. Another way to reduce noise while identifying core accident factors is to combine the process and component ontology, and only the nodes that are part of the final section of the fault tree are important. For example, a worker accidentally activates the press while his hand still in the pressing area will result in pressing action is earlier than intend. One of the core accident factors is press in a deviated state (Early). However the factors that cause the press in such deviated state is the accident path, in this case, a worker accidentally presses the button.

In this example, an additional Accident node is attached to the final Product node. We classify Accident as any deviation from the normal state of raw material, final product, and press action. Accidents will only occur if there are body parts in the press action with raw material or product (deviation of raw material and product in "As well as" or "Other than" state), and from the agent point of view, these should be minimized. More importantly with the classification of any abnormal state as an accident, this will increase the safety requirement and hence more straight constraints in agent design.

A. Importing Accident Data

With 108 accident report feed into the Bayesian Network together with accident classification we mentioned above, we obtain the following result.

Fig. 9 shows the accident probability density function of different actions and materials of worker and press agent. With accident probability density function we can determine the most probable explanation when an accident happened. For example when an accident happened we can see a possible cause will be the Product is in As Well as state (indicating that there is more than just the final product in the press) and that is due to Extract action of Worker agent is in More state, which is due to Press action is in Early state and that is due to Activate action of Worker agent is in Early state.

With accident probability density function it will be valuable for agents to learn the possible accident state and hence choose different actions to avoid results in such a state. However, this accident model is most useful if we combine with actual probability model of any given agent system.

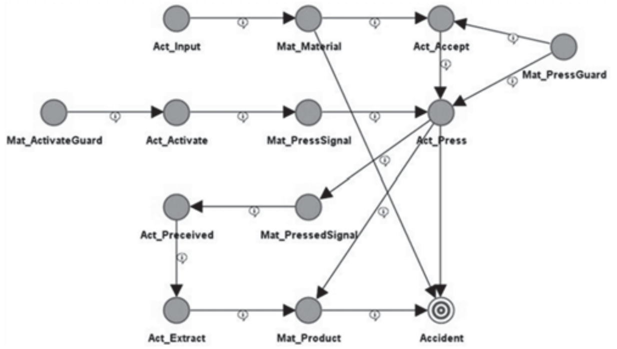


Fig. 8 Bayesian Network with Accident Node

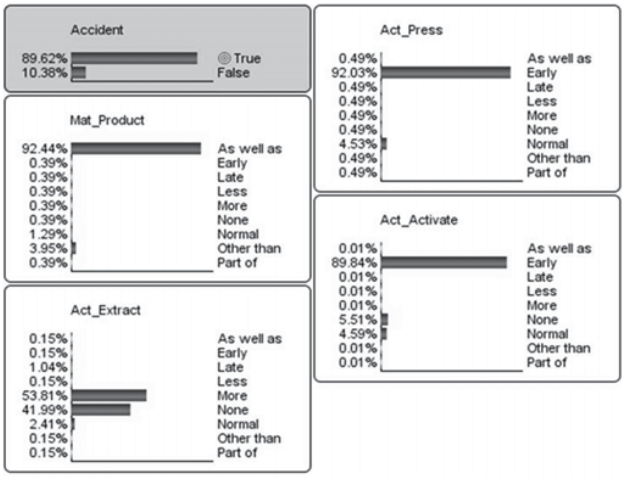


Fig. 9 Accident Probability based on Accident Report

B. Accident Model Combined with Normal Probability Model

To study the general behavior of an agent system, we can use a general probability model of an agent system with minimum safety requirements to calculate the safety constraints of agent's actions. For example, if our minimum safety requirement for this multi-agent press system is 0.5%, then we can use Normal to normal states of different nodes to represent the normal run of the system. From the results below we can see that to achieve 0.5% accident rate the Press action can only have a 0.05% of chances of early, which will form constraints on the agent's action: e.g. Action "Press" of Press agent have following safety constraints:

1. Early state must be less than or equal to 0.05%
2. Normal state must be greater than or equal to 99.74%

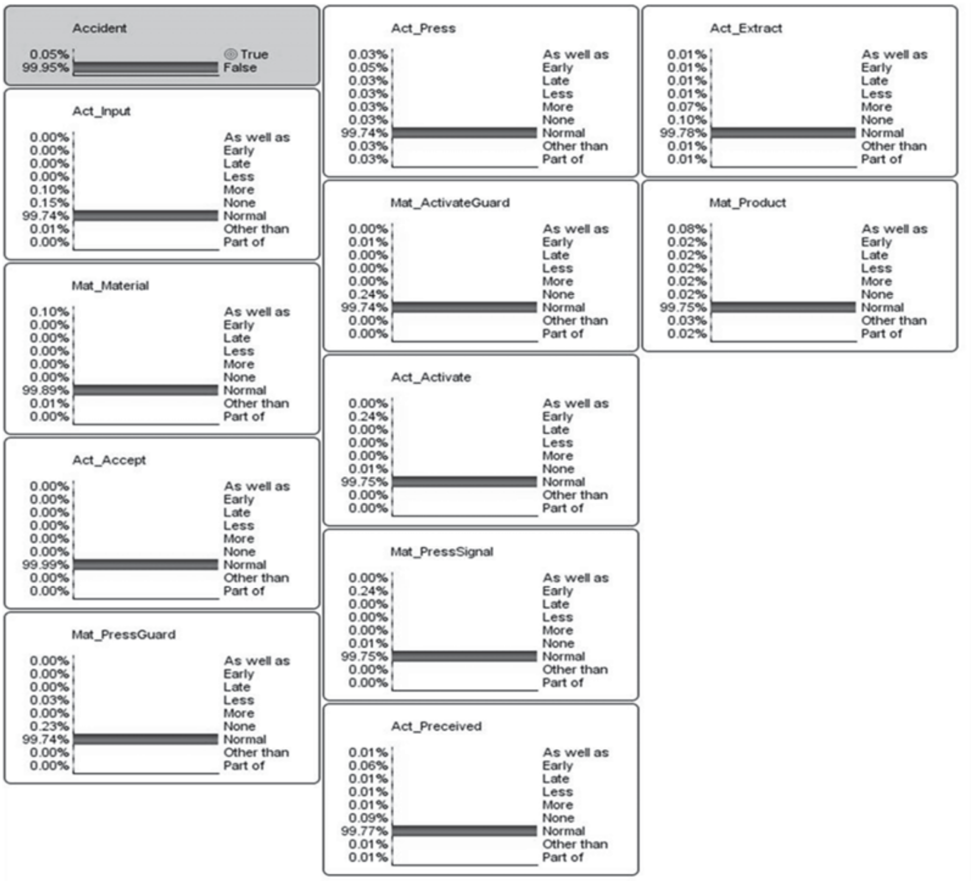


Fig. 10 Accident Probability Based on Safety Requirement

With probability density function we can see the effect on safety levels if the probability of the agent's action is changed. For example, we want to lower the accident rate by introducing a light curtain as Press Guard to activate the Press only if the pressing area is clear from outside intervention, in our case most likely due to workers loading raw material or retrieving the final product. If we look at the Pressing Guard of all accident cases, we can see it either doesn't present or performs below the intended level (Fig. 11).

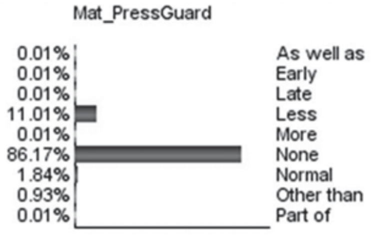


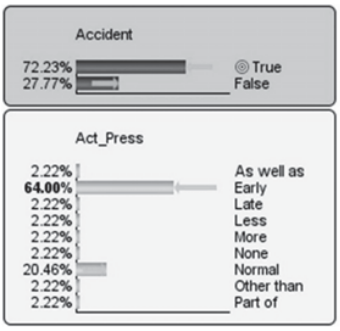
Fig. 11 Probability Density of Pressing Guard

To reduce the accident from happening we need to lower the core cause of the accident and in our example only Pressing action, Raw material and Final product contribute to the core accident. New press guards can only improve the safety record by changing the likelihood of deviation states of these core accident factors. The influence can be expressed in 2 parts:

1. Posterior probability of core accident factors influenced by a given node
2. Probability density function of the different state of the given node.

For example, if the light curtain in Normal state can influence pressing action to lower "Early" state of pressing by 70%, and Normal state of Light curtain contributes 90% of all the possibility of the Pressing Guard, then the actual posterior probability of new press state corrected from "Early" state will be 70%×90%=64%. And by inserting these data back to our Bayesian Network model, we can see the Accident reduce to 72.23% if action "Press" reduce "Early" state to 64%. That is 17.39% of reduction of accident rate from the original. These probabilities will provide guidance and limitations to the design and allow the designer to have a quick check of which design is more cost effective to reduce the accident rate. As a result, to keep safety levels within a given range, it will become an optimization problem of the Bayesian network.

Our safety model can be continually updated with new data and accident reports. If the agent system can record their action, then these data will update the statistical model of their action and material states. Hence, a new probability of accidents can be calculated. Hence, we can constantly adjust the safety constraints of our system.



Accident

72.23% True
27.77% False

Act_Press

2.22% As well as
64.00% Early
2.22% Late
2.22% Less
2.22% More
2.22% None
20.46% Normal
2.22% Other than
2.22% Part of

Fig. 12 Accident Probability Changes

XIV. Discussion

In this paper, we have proposed a way to quantify accidents using previous accident reports by estimating the probabilities of deviations from intended behavior. This allows us to reuse information from different accident reports of the same kind of system but may have a different implementation.

We identify the core accident factors and populate the accident density function from accident event chains. With this accident knowledge model, we can provide a quantified measurement of how safe of a current system if we know the probability transition function of its components. Else for a new system, we provide a way to estimate the accident level by legal requirement and safety level design goal. We show how to use our accident model to design new components and provide continued learning. Next step in our work will be to combine accident data of different systems to estimate the safety level of a new system.

References

[1] Wooldridge, Michael, Nicholas R. Jennings, and David Kinny. "The Gaia methodology for agent-oriented analysis and design." Autonomous Agents and multi-agent systems 3.3 (2000): 285-312.

[2] Padgham, Lin, and Michael Winikoff. "Prometheus: A methodology for developing intelligent agents." Agent-oriented software engineering III. Springer Berlin Heidelberg, 2003. 174-185.

[3] Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. JAAMAS 8(3), 203–236 (2004)

[4] Jamont, Jean-Paul, Clément Raievsky, and Michel Occello. "Handling safety-related non-functional requirements in embedded multi-agent system design." Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection. Springer International Publishing, 2014. 159-170.

[5] Lakner, Rozália, et al. "Agent-based diagnosis for granulation processes." Computer Aided Chemical Engineering 21 (2006): 1443-1448.

[6] Ebrahimipour, V., K. Rezaie, and S. Shokravi. "Enhanced fmea by multi-agent engineering fipa based system to analyze failures." Reliability and Maintainability Symposium (RAMS), 2010 Proceedings-Annual. IEEE, 2010.

[7] Rodríguez-Fernández, Carlos, and Jorge Jesús Gómez-Sanz. "Self-management capability requirements with SelfMML & INGENIAS to attain self-organising behaviours." Proceedings of the second international workshop on Self-organizing architectures. ACM, 2010.

[8] Sterling, Leon, and Kuldar Taveter. The art of agent-oriented modeling. MIT Press, 2009.

[9] Fuxman, A., R. Kazhamiakin, M. Pistore, and M. Roveri. "Formal Tropos: language and semantics." *University of Trento and IRST* 55 (2003): 123.

[10] Borst, Pim, Jan Benjamin, Bob Wielinga, and Hans Akkermans. "An application of ontology construction." In *Proc. of ECAI-96 Workshop on Ontological Engineering. Budapest.* 1996.

[11] Nielsen, Thomas Dyhre, and Finn Verner Jensen. Bayesian networks and decision graphs. *Springer Science & Business Media,* 2009.

[12] Leveson, Nancy. Engineering a Safer World. MIT Press, 2012.

[13] Ministry of Defence, Defence Standard 00-58, Great Britain. *HAZOP Studies on Systems Containing Programmable Electronics*, 2 edition, May 2000.

[14] J. McDermid and T. Kelly, "*Industrial Press: Safety Case*," High Integrity Systems Engineering Group, University of York 1996.

[15] S. Basnyat, N. Chozos, C. Johnson, and P. Palanque. Incident and accident investigation techniques to inform model based design of safety critical interactive systems. In M. Harrison, editor, *Design, Specification and Verification of Interactive Systems 2005*, pages 51–66, Berlin, Germany, 2006. Springer Verlag. Lecture Notes in Computing Science 3941.