Risk Modelling and Reasoning in Goal Models

Yudistira Asnar, Paolo Giorgini, and John Mylopoulos

February 2006

Technical Report # DIT-06-008

Risk Modelling and Reasoning in Goal Models

Yudistira Asnar
University of Trento, Italy
yudis.asnar@dit.unitn.it

Paolo Giorgini
University of Trento, Italy
paolo.giorgini@unitn.it

John Mylopoulos
University of Trento, Italy
University of Toronto, Canada
jm@cs.toronto.edu

Abstract

In software engineering, risks are usually considered and analysed during, or even after, the design of the system. This approach can lead to the problem of accommodating necessary countermeasures in an existing design and possible to reconsider the initial requirements of the system. In this paper, we propose a goal-oriented approach for modelling and reasoning about risks at requirements level. Risks are introduced and analysed along the stakeholders' goals and countermeasures are imposed as part of the requirements of the system-to-be. The proposed framework is based on the Tropos methodology and extends the formal framework with new concepts and qualitative reasoning mechanisms to consider risks since the early phases of the requirements analysis. The risk analysis process is presented and illustrated with some experimental results.

1 Introduction

Traditionally, in software engineering risk analysis is used to identify the high risk elements of the project and provide ways of documenting the impacts of risk mitigation strategies [15]. Moreover, risk analysis has been shown important in the software design phase to evaluate criticality of the system [3]. Risks are analysed and necessary countermeasures are introduced as new functionality of the software. In order to accommodate these new functionality, the approach envisages the necessity of revising the entire design and possible the initial requirements of the system. This can introduce, however, new problems and then lead to system vulnerabilities. Considering risks since the early phases of the software development process can prevent such problems and, as effect, contain the costs of the project. In particular, analysing risks along the stakeholders' needs and objectives, namely before the definition of the requirements of the software, can introduce good criteria for the analyst to evaluate and choose among different alternatives.

Goal-oriented requirement engineering is an emerging research area where the main idea is to put emphasis on *why* certain requirements are needed before analysing *how* they can be realised. This approach facilitates the analyst to understand the real goals of the stakeholders and so evaluate the different alternatives for their satisfaction. Several methodologies and frameworks have been presented in literature, such as KAOS [16], *i** [19], GBRAM [2] and Tropos [4].

Tropos adopts the *i** modelling framework for analysing requirements. During early requirements analysis, the requirements engineer identifies the domain stakeholders and models them as social actors, who depend on one another for goals to be fulfilled, tasks to be performed, and resources to be furnished. Through these dependencies, one can answer why questions, besides what and how, regarding system functionality. Answers to why questions ultimately link system functionality to stakeholder needs, preferences and objectives. The methodology analyses goals by a refinement process in which each goal is decomposed into subgoals and positive/negative contributions are established among goals. So for example, the goal of the production manager to *reduce costs* can be OR-decomposed in *use raw materials effectively* and *reduce labour costs*, and the goal to *have an efficient vehicles production plan* contributes positively to the satisfaction of the goal *reduce costs*.

Through goal models, the analyst can analyse alternative solutions for the satisfaction of stakeholders' goals and choose among them on the base on specific criteria (e.g., minimum-cost [8]). However, Tropos, as well as the other goal-oriented approaches, does not consider risks during the requirements analysis and it can happen that the cheapest alternative corresponds to the most risky one. For instance, suppose that in order to *reduce labour costs* the production manager can either *buy more efficient machines* or *reduce salaries*. Of course, buying new machines is costly for the company and reducing the salaries seems to be the best alternative. However, the reduction of the salaries closely depends on the labour regulation which is issued by the government and that, in some countries, can change frequently and prevent the possibility of the reduction.

In this paper we propose an extension of the Tropos requirements analysis phase to accommodate risk analysis. We extended the goal model formal framework introducing a three layers model, where risks are related to goals and countermeasures to risks (Section 3). Risks can have an impact on goals and countermeasures produce an effect to the mitigation of the risks. A risk analysis process is also discussed and algorithms for the qualitative analysis are presented (Section 4). The developed CASE tool is discussed and shown by experimental results (Section 5). We finally conclude the paper with related work (Section 6) and a final discussion (Section 7).

2 Tropos Goal Analysis

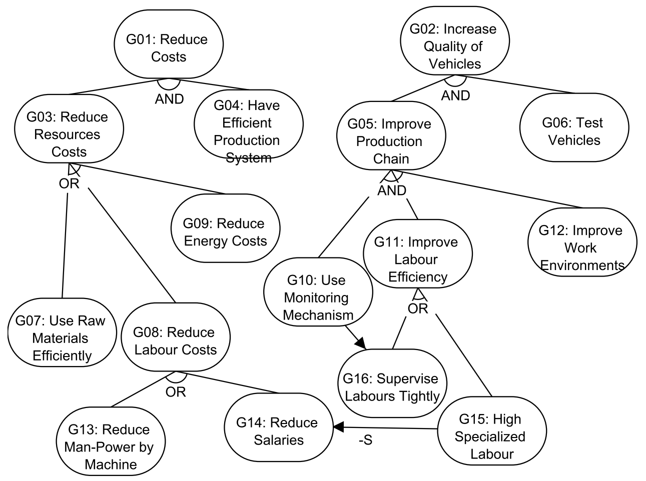
Tropos proposes a formal framework to do requirement analysis that results in a number of goal models represented as graphs $\langle \mathcal{G}, \mathcal{R} \rangle$, where $\mathcal{G}$ are goals and $\mathcal{R}$ are relations (decomposition or contribution relations). If $(G_1, ..., G_n) \overset{r}{\longmapsto} G$ is one of the goal relations in $\mathcal{R}$, $G_1, \ldots, G_n$ are called as the *source nodes* and G is the *target node* of relation r.

Each goal has two attributes SAT-$Sat(G)$ and DEN-$Den(G)$, which quantify the value of evidence for the goal being satisfied and denied, respectively. The values of the attributes are qualitatively divided in the range of $\{(F)ull, (P)artial, (N)one\}$. The attribute is indicated as goal label and is represented by 6 different satisfaction predicates:

- $FS(G)$, $FD(G)$: there is (at least) $full$ evidence that goal G is satisfied (or denied);
- $FS(G)$, $PD(G)$: there is (at least) $partial$ evidence that goal G is satisfied (or denied);
- $NS(G)$, $ND(G)$: there is $none$ evidence that goal G is satisfied (or denied). They are the same with T predicate in [7]. It is not mandatory to write these predicates in formalisation; they could leave implicitly.

The predicates state that there is *at least* a given level of evidence that the goal is satisfied (or denied), and we assume that $FS(G) \geq PS(G) \geq NS(G)$ and $FD(G) \geq PD(G) \geq ND(G)$, with the intended meaning $x \geq y \leftrightarrow x \rightarrow y$.

Qualitative goal analysis in Tropos starts with a number of top goals and each of them is refined by decomposition (AND or OR) into subgoals. For example, consider to model the strategic objectives of a *production department in a vehicles company* and suppose to have as main goals



G01: Reduce Costs
AND
G02: Increase Quality of Vehicles
AND
G03: Reduce Resources Costs
G04: Have Efficient Production System
G05: Improve Production Chain
G06: Test Vehicles
OR
G09: Reduce Energy Costs
AND
G11: Improve Labour Efficiency
G12: Improve Work Environments
G10: Use Monitoring Mechanism
OR
G07: Use Raw Materials Efficiently
G08: Reduce Labour Costs
G16: Supervise Labours Tightly
OR
G13: Reduce Man-Power by Machine
G14: Reduce Salaries
-S
G15: High Specialized Labour

Figure 1. Goal Model for the Vehicle Production Department

to *reduce costs* and to *increase quality of vehicles* (Fig. 1). The goal *reduce costs* (G_1) is AND-decomposed into *reduce resources costs* (G_3) and *have efficient production system* (G_4). Next, *reduce resources costs* (G_3) can be OR-decomposed into *use raw materials effectively* (G_7), *reduce labour costs* (G_8), and *reduce energy costs* (G_9), moreover the goal *reduce labour costs* (G_8) can be satisfied either by *reduce man-power* (G_{13}) or *reduce salaries* (G_{14}). This decomposition and refinements will continue until the goals are not considered tangible goals, i.e., when there is an actor that can fulfil the goal.

Moreover, Tropos goal analysis allows us to model the influence of the satisfaction (denial) of a goal to the satisfaction (denial) of other goals. This influence can be positive or negative and is graphically indicated by "+/−" contribution relations. Tropos also has "++" and "−−" to express *strong positive contribution* and *strong negative contribution*, respectively. For example, the goal *use monitoring mechanism* (G_{10}) in the production system will help in enforcing ("+" contribution) the goal of supervising labour (G_{16}).

There are situations in which we need to model only the influence of the satisfaction of a goal on the denial (or the satisfaction) of another goal. For instance, having *high specialised labour* (G_{15}) impacts negatively to the satisfaction of the goal *reduce salaries* (G_{14}), but we cannot say anything about what happen if the goal *high specialised labour* is denied. In other words we want to separate the effects of SAT and DEN of the source node on the SAT and DEN of the target node. To model such situations, Tropos proposes to use different type of contribution links, namely $-_S$,$+_S$, $-_D$, and $+_D$. In the example, once G_{15} is satisfied, G_{14}

has partial evidence of being denied. Conversely, the denial of G_{15} does not infer that G_{14} will have partial evidence of being satisfied.

The semantics of Tropos goal models is expressed by a set of basic axioms in [7]. For instance, the axioms state that full satisfiability (or deniability) implies partial satisfiability (or deniability); for an "and" relation implies that the full and partial satisfiability of the target node require respectively the full and partial satisfiability of all the source nodes; for a "$+_S$" relation, the axiom states that only the partial satisfiability (but not the full satisfiability) propagates through a "$+_S$" relation. Thus, e.g., an "and" relation propagates the minimum satisfiability value (and the maximum deniability one), while a "$+_S$" relation propagates at most a partial satisfiability value. To this extent, a "$+_S$" relation can be seen as an "and" relation with an unknown partially satisfiable goal. Similar considerations hold for the other relations. The axioms show also the relations monotonically increase the values of satisfaction and denial among goals, namely SAT and DEN of sources nodes monotonically increase SAT and DEN of target nodes. Maximum function is used to combine different contributions.

Tropos provides also two forms of qualitative reasoning[1] on goal models: *forward* [7] and *backward* [8] reasoning. Forward reasoning starts with assigning initial goal labels (SAT and DEN) to a set of goals (typically leaf goals), then propagates the labels to the other goals of the model following the relations of the axioms. This form of reasoning allow us to understand the effects of a particular goal assignment over the whole model. It is useful for example to evaluate whether the assignment allows us to satisfy the stakeholders' top goals or not. On the other hand, backward reasoning works in the opposite direction. The reasoning starts defining desired values of evidence for a number of goals (typically top goals) and constraints (i.e., the level of conflict that is accepted for the goal model, minimum evidence value of certain goal), then the reasoning tries to find an assignment for leaf goals (input goals) that can satisfy the desired values. The reasoning is enhanced by a number of criteria (e.g, minimum-cost [8]) to evaluate and choose among all the possible solutions. Backward reasoning is useful to find a possible assignment for leaf goals that satisfies the stakeholders' top goals.

3 Extending Goal Model for Risk Management

As said, Tropos does not consider external events in the requirements analysis phase and it is not possible to analyse the effects of unpredictable situations on the stakeholders' goals. In this section, we extend the Tropos goal model introducing the new primitives *event* and *treatment*. Roughly, a risk is an event that has a negative impact on the satisfaction of a goal, while a treatment is a countermeasure that can be adopted in order to mitigate the effects of the risk.

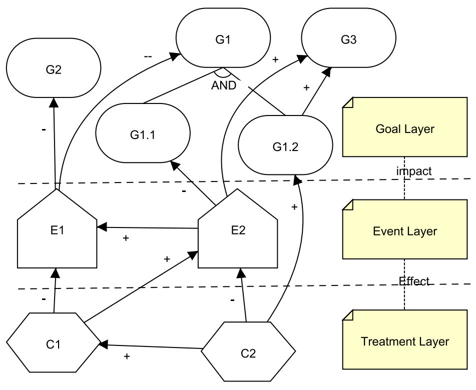


Figure 2. Extended Goal Model

In our framework we adopt the WordNet[2] definition for event:

- something that happens at a given place and time;
- a special set of circumstances;
- a phenomenon located at a single point in space-time;
- a consequence; i.e., a phenomenon that follows and is caused by some previous phenomena.

An event becomes a risk when it produces a negative effect, whereas it is an opportunity when it produces positive effects. Probabilistic Risk Assessment (PRA) [12] has identified two mandatory properties for risk: likelihood and severity. Conversely, in our framework we consider likelihood as a property of the event, whereas severity/impact is introduced as a contribution relation (negative/positive) between an event and a goal. This allows us to model situations where a single event impacts on more than one goal.

Adopting the idea of three-layers analysis of Defect Detection and Prevention (DDP) [6], we consider three different layers in goal models Fig. 2. The strategic interests of the stakeholders are modelled in the first-layer (goal layer) using the classical Tropos goal model approach. Subsequently, risks and opportunities that result relevant for the goal layer, are analysed in the second-layer (event layer), whereas countermeasures to mitigate risks are introduced and analysed in the third-layer (treatment layer). Graphically, we represent an event as a pentagon (same representation used in Fault Tree Analysis (FTA) [18]) and a treatment

[1]Actually, Tropos proposes also quantitative reasoning mechanisms [7, 8]

[2]http://wordnet.princeton.edu/

as a task (no conceptual distinction between task [4] and treatment). As for goals, events and treatments can be decomposed in sub-events and sub-treatments and related by contribution relations to other goals, events, and treatments. Intra-layer and inter-layer relations [9] are fully adopted in the framework to capture all possible situations. The complete meta-model of the goal model is presented in Fig. 2

3.1 Event Layer

Based on the definitions in WordNet, we resume that an *event* can be a circumstance (e.g., "It rains", "There is a war"), the outcome of the satisfaction of a goal or the outcome of the accomplishment of a task. In our framework, a *risk* is defined as an uncertain event with negative impact. This notion is slightly different from *threat* [13] in computer security and hazardous condition in reliability engineering [11], which are only defined as a potential circumstance that could cause harm or loss and not specifying the notion of likelihood.

Events can be identified applying different approaches, such as obstacle analysis in KAOS [17], Taxonomy-base risk identification [5], or Risk in Finance [10]. Afterwards, an event is analysed by a decomposition into sub-events until each leaf event can be considered an independent event. Leaf events are used later to find proper countermeasures.

An event can influence more than one goal. So for example, in Fig. 3 the event *strike* (E_5) obstructs the satisfaction of *reducing salaries* (G_{14}) because in this circumstance labours can demand an increment of the salary. On the other hand, it also obstructs the goal *improve production chain* (G_5) since it can compromise and slow down the production. An event can be considered as a risk for certain goals and at the same time an opportunity for other goals. For instance, the event have a *new competitor* (E_7) is a risk that obstructs the achievement of the goal *high specialised labour* (G_{15}) because the competitor can offer better conditions to the labour. However, the event can also be seen as an opportunity for the goal *improve work environment* (G_{12}), because it gives more motivations to the employees to compete with other companies.

As we already said an event can be characterised by two properties : *likelihood* and *severity/impact*. **Likelihood** is defined as how likely an event occurs [1]. In our framework, we represent *likelihood* by the level of evidence that supports and prevents the occurrence of the event (SAT and DEN). We adopt for SAT and DEN of an event the same Tropos meaning of SAT and DEN for a goal.

By **impact**, we mean the influence of an event to the goal fulfilment. This definition is similar to the definition given in DDP [6] and to the definition of *severity* in FMECA [1]. We classify impact as follows:

- Strong Positive($++$) - the event occurrence gives a $strong$ contribution of the goal satisfaction;
- Positive($+$) - the event occurrence gives a $fair$ contribution of the goal satisfaction;
- Negative($-$) - the event occurrence gives a $fair$ contribution of the goal denial;
- Strong Negative($--$) - the event occurrence gives a $strong$ contribution of the goal denial.

Since the effect of an event obstructs a goal only when it occurs (i.e., denial of an event does not give any impacts), in our model we use only r_S relations, i.e., $++_S$, $+_S$, $-_S$, and $--_S$, between an event and goals.

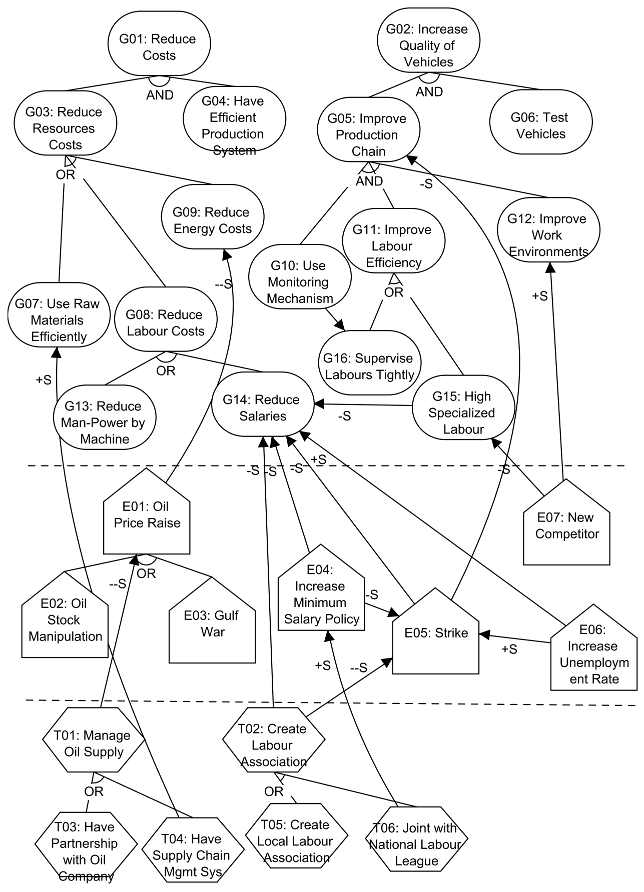

G01: Reduce Costs
G02: Increase Quality of Vehicles
AND
G03: Reduce Resources Costs
G04: Have Efficient Production System
G05: Improve Production Chain
G06: Test Vehicles
OR
G09: Reduce Energy Costs
G10: Use Monitoring Mechanism
G11: Improve Labour Efficiency
G12: Improve Work Environments
G07: Use Raw Materials Efficiently
G08: Reduce Labour Costs
G16: Supervise Labours Tightly
G13: Reduce Man-Power by Machine
G14: Reduce Salaries
G15: High Specialized Labour
E01: Oil Price Raise
E02: Oil Stock Manipulation
E03: Gulf War
E04: Increase Minimum Salary Policy
E05: Strike
E06: Increase Unemployment Rate
E07: New Competitor
T01: Manage Oil Supply
T02: Create Labour Association
T03: Have Partnership with Oil Company
T04: Have Supply Chain Mgmt Sys
T05: Create Local Labour Association
T06: Joint with National Labour League
+S
-S
--S

Figure 3. Extended Goal Model for the Vehicle Production Department

3.2 Treatment Layer

Once the events have been analysed, the analyst identifies and analyses the countermeasures to be adopted in order to mitigate the risks. The mitigation of a risk can be realised in two different ways: reducing the *likelihood* or reducing the *impact*. However, in Tropos goal model it is not possible to model a relation between a node and a relation (only between nodes), so in this paper we do not consider the reduction of the impact as a possible mitigation. This will be part of our future work.

Similarly to goals and events, for countermeasures we use SAT and DEN to represent the evidence that supports and prevents the action. A countermeasure has effect on the event layer, and in particular over risks. We represent the **effect** of a countermeasure as a relation, where its strength is expressed by the sign of the contribution relations. As for events, we are interested to the propagation of the evidence for the success of a countermeasure (SAT) and therefore we limit the relations between countermeasures and events to r_S relations. A countermeasure mitigates a risk when it adds (propagates) evidence for its denial.

In our model we also allow for relations between the treatment layer and the goal layer. This is useful to model situations where a countermeasure adopted to mitigate a risk has also a contribution (especially negative) to some goals. For instance in Fig. 3, the countermeasure *create a labour association* (T_2) can mitigate the likelihood of the event *strike* (E_5) – of course this is not always true. However, the association can have a better bargaining power w.r.t. the individual worker and obtain an increment of the salaries. This produces a negative effect on the satisfaction of the goal *reduce salaries* (G_{14}).

3.3 Meta-model

Table 1 resumes all the possible relations among goals, events, and treatments. For instance, the possible relations form an event to other events are decomposition and SAT contribution relations. Basically, all objects (Goal, Event, Treatment/Task) can be operated with all type of decomposition and contribution relations among the objects in the same layer. Moreover, the only relation that can be used across the layers is the contribution relation. For events and treatments, there are limitations in the type of the contribution relations (only SAT-Relation; $r_S \in \{++_S, +_S, --_S, -_S\}$) as shown in the meta-model represented in Fig. 4. In the following we describe some situations that could be modelled with cross layer contribution relations [3]:

goal → event to model that a goal increases/reduces the occurrence of an event. For instance, the goal *give big annual bonus* can reduce the likelihood of *strike*;

goal → countermeasure to model that a goal supports/prevents the countermeasure accomplishment. For instance, the achievement of the goal *have a close partnership with the oil company* helps to *manage oil supply* in the company;

event → goal to model risk, namely the impact of event on the goal satisfaction. For instance, the event *oil price raise* can obstruct the satisfaction of the goal *reduce energy costs*;

event → treatment to model the influence of the occurrence of an event on the countermeasure accomplishment. For example, the event *gulf war* can make the oil very precious s.t. the oil company feels no need to *have partnership* with any end-customers;

treatment → goal to model the side effect (negative or positive) of a countermeasure on the goal layer. For example, the countermeasure *have a supply management system* has a positive impact to the satisfaction of the goal *use raw materials efficiently*.

treatment → event to model the effect of a countermeasure to the mitigation of a risk. For example, having a good *managing oil supply* can mitigate the risk *oil price raise*;

[3]Including *impact-contribution* and *effect-contribution*

Source \ **Target**	*Goal*	*Event*	*Treatment*
Goal	Decomp., Cont.	Cont.	Cont.
Event	SAT Cont.	Decomp., SAT Cont.	SAT Cont.
Treatment	SAT Cont.	SAT Cont.	Decomp., Cont.

Table 1. Relations in the Extended Goal Model

Decomposition $r \in \{and, or\}$, Contribution $r \in \{++, +, --, -\}$, SAT Contribution $r_S \in \{++_S, +_S, --_S, -_S\}$

4 Risk Analysis

As already discussed in the introduction, we intend to support the analyst in evaluating different requirements alternatives with respect to risk. In this section, we describe the methodological process and the qualitative reasoning mechanisms that are used in the different steps of the process.

The analysis process is described in the Algorithm 1 and consists of the following three steps:

1. find the alternative solutions (line 2-3),
2. evaluate each alternative against the relevant risks (line 8-10)
3. assess the countermeasures to mitigate the risks (line 11-16).

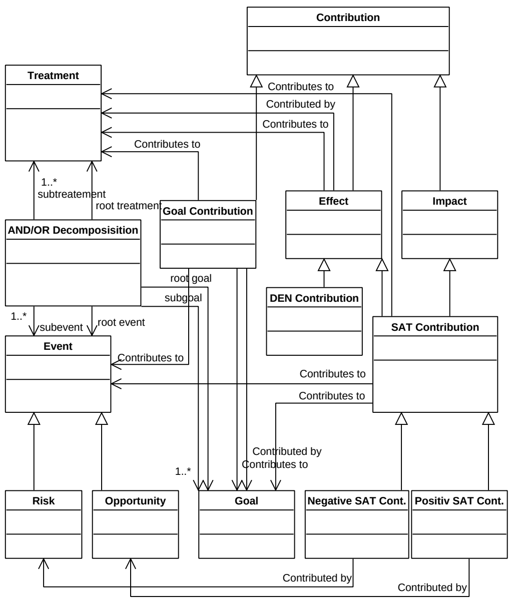


Figure 4. Meta-model of the Extended Goal Model

The process starts taking in input the extended goal model and a set of desired values for top goals (i.e., satisfaction values-SAT and acceptable risk values-DEN), and a number of goals as possible candidates for the final solution (input goals). For instance, we may desire to fully satisfy G_1 ($Sat(G_1)$=F) without any risk ($Den(G_1)$=N).

$Backward\_Reasoning$ (line 2) generates a set of possible assignment values for the input goals that can satisfy the desired values. We use the standard Tropos backward reasoning limited to the goal layer (i.e., not considering the relations with the other two layers) and not considering any constraints. So for instance, in order to fully satisfy G_1 we can either fully satisfy $\{G_7, G_4\}$, $\{G_{13}, G_4\}$ or $\{G_{14}, G_4\}$.

The analyst chooses a subset of the alternatives on the basis of a certain criteria (i.e., minimum-cost [8], softgoals) called $candidate\_solution$ (line 3). The rest of the process will be limited to the analysis of this subset. For instance, the analyst can decide to choose the alternatives $\{G_{13}, G_4\}$ and $\{G_{14}, G_4\}$ considering the goal *use raw materials* (G_7) is too expensive.

Each $candidate\_solution$ is now evaluated against risks and then necessary countermeasures are introduced (line 4-18). First, the analyst checks whether the $candidate\_solution$ needs countermeasures to obtain the desired values in the top goals. If not the $candidate\_solution$ is added directly to the $solution$ and its cost is calculated (line 6). Otherwise, countermeasures must be introduced in the $candidate\_solution$ (line 8-16). So for example, evaluating $\{G_{13}, G_4\}$ we see from Fig. 3 that there is no risk associated to them so we do not need any countermeasure. It is different for $\{G_{14}, G_4\}$.

In order to define the countermeasures, we first need to calculate the risk values that are acceptable for the stakeholders. In other words, we need to find the maximum assignable values of risk that produce an acceptable DEN value for top goals. So for example, in our example we need to find a set of countermeasures able to mitigate risks s.t. $Den(G_1)$=N. To do this we need to consider only the relevant risks for the $candidate\_solution$, namely risks that have an impact on the input goals and goals reachable from them by decomposition ("related goals"). For instance, since the related goals of $\{G_{14}, G_4\}$ are $\{G_{14}, G_8, G_3, G_4, G_1\}$ we consider only the risks $\{E_4, E_5\}$.

Algorithm 1 Risk_Analysis Process

```
Ensure: analyse risk for each alternative solutions and find necessary
    countermeasures to ensure the satisfaction of top goals.
Require: goal_model $\langle \mathcal{G}, \mathcal{R} \rangle$ , label_array top_goals, node_array input_goals, label_array events
 1: solution_array solution {solution that has already encompassed risks
    and necessary countermeasures}
 2: alt_solution ←$Backward\_Reasoning(\langle \mathcal{G}, \mathcal{R} \rangle$ , nil,
    $top\_goals, input\_goals)$
 3: candidate_solution ←$Select\_Can\_Solution(candidate\_solution)$
    {alt_solution ⊆ candidate_solution}
 4: for all $S_i \in candidate\_solution$ do
 5:   if $Satisfy(\langle \mathcal{G}, \mathcal{R} \rangle , top\_goals, \langle S_i, events, \mathbf{nil} \rangle)$ then
 6:     add(solution,$\langle S_i, \mathbf{nill}, Calc\_Cost(S_i, \mathbf{nil}) \rangle$)
 7:   else
 8:     boolean_array Related_Goals ←
          $Related\_Goals(\langle \mathcal{G}, \mathcal{R} \rangle , S_i)$
 9:     labels ←$Standard\_Forward\_Reasoning(\langle \mathcal{G}, \mathcal{R} \rangle , S_i)$
10:     acc_events ←$Calc\_Event(labels, related\_goals, events)$
11:     nec_treatment ←$Backward\_Reasoning(\langle \mathcal{G}, \mathcal{R} \rangle , events,$
          $acc\_events, avail\_treatment)$
12:     for all $T_j \in nec\_treatment$ do
13:       if $Satisfy(\langle \mathcal{G}, \mathcal{R} \rangle , top\_goals, \langle S_i, events, T_j \rangle)$ then
14:         add(solution,$\langle S_i, T_j, Calc\_Cost(S_i, T_j) \rangle$)
15:       end if
16:     end for
17:   end if
18: end for
```

$Standard\_Forward\_Reasoning$ (line 9) is used to propagate the input values of the $candidate\_solution$ in the model and so evaluate the impact of the risk. Once we have an assignment for all the goals of the model we can calculate the acceptable values for the event (i.e., values that can still satisfy the desired top goals). This is done by the $Calc\_Event$ Algorithm 5. We illustrate it later.

Backward reasoning is again applied (line 11) to find possible treatments that can guarantee the acceptable risk (acc_event). Of course, we can have more than one possible combination of treatments. For instance, defining $Sat(E_4)$=P, $Den(E_4)$=F and $Sat(E_5)$=F, $Den(E_5)$=F as acceptable values for risk, backward reasoning results $\{T_5\}$

$Sat(E)$ \ $Den(E)$	Full	Partial
Full	$Sat(E)$ = N, $Den(E)$ = N	$Sat(E)$ = P, $Den(E)$ = N
Partial	$Sat(E)$ = N, $Den(E)$ = P	$Sat(E)$ = N, $Den(E)$ = N

Table 2. Conflict Resolution for Events

or $\{T_6\}$ as possible treatments.

Finally (line 12-16), each set of treatments are evaluated with respect to the initial desired values (indeed, there could be the case that a treatment has negative contribution directly to the goal layer) and the affordable costs. To do this we use again the $Satisfy$ algorithm (line 13) that basically consists of the following three steps:

1. the input values for goal, events and treatments are propagated over the whole model;
2. conflicts in the event layer are solved ;
3. the input values for goals, treatments, events (as results of step 2) are propagated excluding incoming relations to events.

For the three steps, we use the $New\_Forward\_Reasoning$ (Algorithm 2), which is a revised version of the algorithm presented in [7]. The main difference is that here we extend the propagation to the event and the treatment layer and we introduce the conflict resolution step. As discussed in the previous section, the influence of a risk on a goal depends on the SAT value of the risk and a mitigation corresponds to the increment of the level of the DEN. For example, the impact of E_5 on the goal G_{14} depends on the SAT value (e.g.,$Sat(E_5)$=F), and the effect of T_5 on G_{14} is to increase the DEN value of E_5 (e.g., $Den(E_5)$=F). A countermeasure is effective when it is able to generate a conflict between SAT and DEN in the risk. For example, after taking T_5 there is a conflict in E_5 (i.e., $Sat(E_5)$=F and $Den(E_5)$=F). The conflict resolution step allows us to separate the effects of the countermeasures on the goal layer from the impact of the risks. Table 2 presents the conflict resolution rules we adopted.

Algorithm 2 New_Forward_Reasoning

```
Ensure: propagate evidence to the goal_model
Require: goal_model $\langle \mathcal{G}, \mathcal{R} \rangle$ , label_array initial
 1: resolution ←false
 2: pre_res ←
        $Label\_Propagation(\langle \mathcal{G}, \mathcal{R} \rangle, initial, resolution)$
 3: post_res ←$Resolve\_Conflict(\langle \mathcal{G}, \mathcal{R} \rangle, initial, pre\_res)$
 4: resolution ←true
 5: result ←$Label\_Propagation(\langle \mathcal{G}, \mathcal{R} \rangle, post\_res, resolution)$
 6: return result
```

$Calc\_Event$ algorithm (Algorithm 5) basically does the reverse of conflict resolution and distinguishes between risk

Algorithm 3 Label_Propagation

```
Require: goal_model $\langle \mathcal{G}, \mathcal{R} \rangle$ , label_array initial, boolean resolution
 1: label_array current, old
 2: current = initial
 3: while $old \neq current$ do
 4:     old ←current
 5:     for all $N_i \in \mathcal{G}$ do
 6:         if not ($resolution$ and $Is\_Event(N_i)$) then {checking
            whether conflict resolution has already applied at events or not}
 7:             current[i] ←$Update\_Label(i, \langle \mathcal{G}, \mathcal{R} \rangle, old)$
 8:         end if
 9:     end for
10: end while
11: return current
```

Algorithm 4 Update_Label

```
Require: int i, goal_model $\langle \mathcal{G}, \mathcal{R} \rangle$ , label_array old
 1: for all $R_j \in \mathcal{R}$ s.t. $target(R_j) = N_i$ do
 2:     sat$_{ij}$ = $Apply\_Rules\_Sat(N_i, R_j, Old)$
 3:     den$_{ij}$ = $Apply\_Rules\_Den(N_i, R_j, Old)$
 4: end for
 5: return $\{max(max\_array(sat_{ij}), Old[i].sat)$,
    $max(max\_array(den_{ij}), Old[i].den)\}$
```

Algorithm 5 Calc_Event

```
Ensure: calculate acceptable value of risks s.t the input goals can satisfy
    top goals
Require: label_array goals, label_array events, boolean_array related_goals
 1: acceptable_events ←events
 2: for all $B_i \in related\_goals$ do
 3:     if $B_i$ then
 4:         for all ($R_j \in \mathcal{R}$ s.t. $target(R_j) = goals_i$) and
            $Is\_Event(source(R_j))$ do
 5:             node←$source(R_j)$
 6:             k ←$Node\_Index(source(R_j))$
 7:             if ($R_j \in \{-_S, --_S\}$) and $goals_i.den < node.sat$ then
                {risk}
 8:                 if $goals_i.den = N$ and $node.sat = F$ then
 9:                     acceptable_events$_k$.den←F
10:                 else if ($goals_i.den = N$ and $node.sat = P$) or
                    ($goals_i.den = P$ and $node.sat = F$) then
11:                     acceptable_events$_k$.den←P
12:                 end if
13:             else if ($R_j \in \{+_S, ++_S\}$) and $goals_i.sat > node.sat$
                then {opportunity}
14:                 if $goals_i.sat = P$ and $node.sat = N$ then
15:                     acceptable_events$_k$.sat←P
16:                 else if $goals_i.sat = F$ then
17:                     acceptable_events$_k$.sat←F
18:                 end if
19:             end if
20:         end for
21:     end if
22: end for
23: return acceptable_events
```

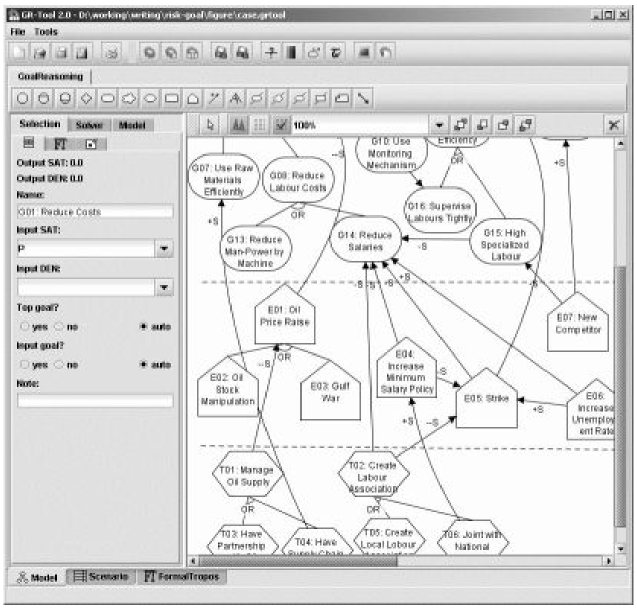


Figure 5. Risk Analysis Tool

and opportunity events, line (8-12) and line(14-18), respectively. For example, since we want no risk on G_{14} ($Den(G_{14})$=N) but E_4 andE_5 produce a $partial$ denial, we need to introduce countermeasures so create a conflict s.t. E_4 and E_5 do not impact on G_{14}. In this case, $Calc\_Event$ produces the acceptable values of risk $Den(E_4)$=P and $Den(E_5)$=F to neutralise the $Sat(E_4)$=P and $Sat(E_5)$=F.

As final result of the process we obtain all the possible solutions (among the $candidate\_solution$) that can satisfy the initial requirements and costs associated to them. Now, the analyst can decide which solution to adopt.

5 Tool and Experimental Results

In this section, we briefly present the tool we have developed and some experimental results obtained with it.

The tool is an extension of the Goal Reasoning Tool[4] (GR-Tool) developed within the Tropos project. Basically, the tool (Fig. 5) is graphical tool in which it is possible to draw the extended goal models and run the algorithms and tools for (standard and new) forward and backward reasoning. The algorithms presented in Section 4 have been fully developed in JAVA and are embedded in the tool. For their implementation we started from the GR-Tool reasoning mechanisms and we have re-implemented them introducing the necessary modifications as described in Section 4. For more details about the GR-Tool and its extension we suggest to visit the Tropos web page.

[4]http://sesa.dit.unitn.it/goaleditor/

Input Goal	Cost	S1	S2	S3	S4	S5	S6	S7	S8
G04: Have Eff. Prod. Sys.	6	X	X	X	X	X	X	X	X
G06: Test Vehicles	4	X	X	X	X	X	X	X	X
G07: Use Raw Materials Eff.	4.5			X			X		
G09: Reduce Energy Costs	3	X			X				
G10: Use Mon. Mechanism	5	X	X	X	X	X	X	X	X
G12: Improve Work Env.	6	X	X	X	X	X	X	X	X
G13: Reduce Man-Pow. by Mach.	9							X	X
G14: Reduce Salaries	3.5		X			X			
G15: High Specialized Lab.	7				X	X	X		X
G16: Supervise Lab. Tightly	3.5	X	X	X				X	
Total Cost		27.5	28	29	31	31.5	32.5	33.5	37

Table 3. Cost of Alternative Solutions

To test our approach and its implementation we run a number of experiments for the case of the *Production Department of a Vehicles Company* presented in Fig. 3.

Table 3-5 summarise an example of these results. Suppose we want to obtain a partial satisfaction of goal G_1 (*reducing cost*), a fully satisfaction of G_2 (*increasing quality of vehicles*), and avoid any risk on them. So we have as input $\{Sat(G_1)$=P, $Sat(G_2)$=F, $Den(G_1)$=N, $Den(G_2)$=N$\}$. Executing $Backward\_Reasoning$ we find a set of possible solutions (as reported in Table 3) and we select among them the cheapest one S1. The total cost of each solution is calculated summing up the costs of the single input goals. Of course, other criteria can be adopted for the selection of the solution. As discussed in [8], for instance we could also evaluate the effects of the solution over some qualities (softgoals). Forward reasoning is applied then to calculate the effects of the selected solution to the other goals of the model (column “Goal-Out” Table 4).

Now, let suppose we have evidence about the occurrence of some of the events and want to see the impact of them on the goal layer. For example, considering the event assignment reported in column “Event-In” (i.e., $Sat(E_2)$=P, $Sat(E_4)$=P, $Sat(E_5)$=F, $Sat(E_6)$=P, $Sat(E_7)$=P), we obtain that (“Event-Out”) top goals G_1 and G_2 are both partially satisfied and denied. In order to re-obtain the desiderate values for top goals we need to find necessary treatments able to mitigate the risks. There are four possible countermeasure sets that could be taken to mitigate the risks (see Table 5) and the total cost of countermeasures can be calculated summing up the single cost of input treatments. In this experiment, we adopt C2 (i.e., $Sat(T_4)$=P, $Sat(T_5)$=F) based on their costs and their side effects. Even C2 is the most expensive, it is worth enough compared with T_4 positive contribution to G_7 and both treatments do no agitate the risks (conversely, T_6 triggers the risk E_4). Finally, the tool generates the final configuration with input S1 and C2 (in column ”Treatment-Out”) where our desired values for top goals are again obtained.

Let suppose we take the second cheapest solution S2. The “relevant-risks” for this alternative solution are E_4, E_5, E_6, and E_7. Consequently, the countermeasures that are

	Goal			Event			Treat.	
	In	Out		In	Out		Out	
	S	S	D	S	S	D	S	D
E01: Oil Price Raise	-	-	-	-	P	-	-	-
E02: Oil Stock Manipulation	-	-	-	P	P	-	P	-
E03: Gulf War	-	-	-	-	-	-	-	-
E04: Inc. Min. Salary Policy	-	-	-	P	P	-	P	-
E05: Strike	-	-	-	F	F	P	-	-
E06: Inc. Unemployment Rate	-	-	-	P	P	-	P	-
E07: Having Competitor	-	-	-	P	P	-	P	-
G01: Reduce Costs	-	P	-	-	P	P	P	-
G02: Increase Quality of Vehicles	-	F	-	-	F	P	F	-
G03: Reduce Resources Costs	-	P	-	-	P	P	P	-
G04: Have Eff. Prod. System	P	P	-	P	P	-	P	-
G05: Improve Production Chain	-	F	-	-	F	P	F	-
G06: Test Vehicles	F	F	-	F	F	-	F	-
G07: Use Raw Materials Efficiently	-	-	-	-	-	-	P	-
G08: Reduce Labour Costs	-	-	-	-	P	P	P	P
G09: Reduce Energy Costs	P	P	-	P	P	P	P	-
G10: Use Monitoring Mechanism	F	F	-	F	F	-	F	-
G11: Improve Labour Efficiency	-	F	-	-	F	-	F	-
G12: Improve Work Environments	F	F	-	F	F	-	F	-
G13: Reduce Man-Power by Machine	-	-	-	-	-	-	-	-
G14: Reduce Salaries	-	-	-	-	P	P	P	P
G15: High Specialised Labour	-	-	-	-	-	P	-	P
G16: Supervise Labours Tightly	F	F	-	F	F	-	F	-
T01: Manage Oil Supply	-	-	-	-	-	-	P	-
T02: Create Labour Association	-	-	-	-	-	-	F	-
T03: Have Partnership with Oil Comp.	-	-	-	-	-	-	-	-
T04: Have Supply Chain Mgmt Sys	-	-	-	-	-	-	P	-
T05: Create Local Labour Association	-	-	-	-	-	-	F	-
T06: Joint with National Labour League	-	-	-	-	-	-	-	-

Table 4. SAT-DEN **values in Risk Analysis of S1**

Treatment Input	Cost	S1				S2	
		C1	C2	C3	C4	C5	C6
T03: Have Partnership with Oil Comp.	6	X		X			
T04: Have Supply Chain Mgmt Sys	8		X		X		
T05: Create Local Labour Association	6	X	X			X	
T06: Joint with National Labour League	5			X	X		X
Total Cost		12	14	11	13	6	5

Table 5. Cost of Possible Treatments

necessary to be taken are T_5 or T_6. Among the two, even if more costly, T_5 results to be better than T_6 since it does not have any effects on the events' occurrence (T_6 has a "$+_S$" contribution on E_4). Comparing the alternatives, we can conclude taking S2 and C5 is cheaper than S1 and C2 and moreover S2 is less risky than S1 (i.e., the risks of S2 $\subset$ the risks of S1). Similar consideration can be done for the other solutions.

6 Related Work

The most relevant works in literature that have similarities with our approach are KAOS and DDP . We briefly discuss them in the following.

KAOS [16] is the methodology that provides a specification by capturing *why*, *who*, and *when* aspects in addition to *what* aspect in requirements. KAOS, moreover, has already been extended beyond goals, by defining *obstacles* which can be seen as boundaries in requirement analysis. An obstacle is defined as an undesirable behaviour to the strategic interests of stakeholders which can be used to model the risk. In [16], they demonstrate how to derive the obstacles from the goal structure and to ensure the completeness of obstacles with respect to the goal structure and its domain. This technique assumes a complete knowledge to define the domain of the goal, which is hardly to meet in the reality. Conversely, Tropos defines the axioms as "at least" premises which assume there are not complete knowledge to define the number of evidence. The methodology, also, introduces the guidance for defining obstacle resolution to mitigate and, possibly, eliminate the obstacle. Though we can represent a risk as an obstacle, two mandatory properties of risk (likelihood and severity) are still missing. However, the obstacle concept and all related analysis (obstacle identification, obstacle resolution) could be used as a basis for the further enhancement of our framework.

DDP [6] is a model that is developed and applied in Jet Propulsion Lab. and NASA. DDP consists of three layers model (Objectives, Risks, Mitigation). In this model, each objective has a *weight* to represent its importance. A risk has a *likelihood* of occurrence and a mitigation has a *cost* for accomplishment (namely resource consumption). Severity of the risk is represented by impact relation between objective and risk. Moreover, DDP model specifies how to compute the level of objectives achievement and the cost of mitigation[5] from a set of taken mitigation. This calculation allows us to evaluate the impact of taken countermeasures and thus supports the decision making process. The DDP model also allows us to work together with other quantitative tools (FMECA [1], FTA [18]) to model and assess risk/failure. In [6], they demonstrate how to integrate FMECA and FTA into DPP model. There are some differences between the concept of objective [6] and the concept of goal in goal-oriented requirement engineering [4, 16, 19, 2]. In goal-oriented, the goal has a structure (e.g., tree, graph). Conversely, DDP depicts objective (also risk and mitigation) as a solitary object (i.e., there is no relation among objects in the same layer). This feature could limit in modelling situations (e.g., domino effect [14]) in which the occurrence of a risk can increase/decrease the likelihood of other risks. DDP model allows us only to establish relations between layers (i.e., risks to goals, mitigation to risk) which is not enough to represent all possible situations. For example, there could be cases where a treatment affect also the satisfaction of some objective. Our framework adopts and enhances the idea of three-layers model.

7 Conclusions and Future Work

In this paper, we have presented a framework to model and reason about risk within the requirements engineering

[5]objective restoration categories as one kind of mitigation action

process. We have adopt and extended the Tropos goal modelling framework and proposed qualitative reasoning algorithms to analyse risk during the process of evaluation and selection of alternatives. However, our approach has some limitations that we want to overcome as part of our future work. Specifically, as discussed in the paper our modelling framework supports only relationships between nodes (goals, events, tasks) and this could be a limitation when we want to model situations where a treatment mitigates the risk reducing its impact on the goal layer. In other words, we need to extend the framework introducing the possibility to establish relations also between nodes and arcs. Another limitation is related to the fact that in our framework the values for SAT and DEN can only increase monotonically and it is not possible to model the fact that an event decreases the evidence for the satisfaction of a goal. Finally, as done for goal models we want to propose also quantitative reasoning mechanisms where evidence is expressed in term of probability.

Acknowledgement

This work has been partly supported by the projects EU-SENSORIA, EU-SERENITY, FIRB-ASTRO, PAT-MOSTRO, PAT-STAMPS.

References

[1] *Military Standard, Procedures for Performing a Failure Mode, Effects, and Critical Analysis (MIL-STD-1692A)*. US Department of Defense, 1980.

[2] A. I. Anton. Goal-Based Requirements Analysis. In *ICRE '96: Proceedings of the 2nd International Conference on Requirements Engineering*, page 136, Washington, DC, USA, 1996. IEEE Computer Society.

[3] B. W. Boehm. Software Risk Management: Principles and Practices. *IEEE Software*, 8(1):32–41, 1991.

[4] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[5] M. J. Carr, S. L. Konda, I. Monarch, and F. C. Ulrich. Taxonomy-Based Risk Identification. Technical Report CMU/SEI-93-TR-6, ESC-TR-93-183, Software Engineering Institute, Carnegie Mellon University, June 1993.

[6] M. S. Feather. Towards a Unified Approach to the Representation of, and Reasoning with, Probabilistic Risk Information about Software and its System Interface. In *15th IEEE International Symposium on Software Reliability Engineering*, pages 391–402. IEEE Computer Society, November 2004.

[7] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Formal Reasoning Techniques for Goal Models. *Journal of Data Semantics*, October 2003.

[8] P. Giorgini, J. Mylopoulos, and R. Sebastiani. Simple and Minimum-Cost Satisfiability for Goal Models. In *CAISE '04: In Proceedings International Conference on Advanced Information Systems Engineering*, volume 3084, pages 20–33. Springer, June 2004.

[9] P. Giorgini, J. Mylopoulos, and R. Sebastiani. Goal-Oriented Requirements Analysis and Reasoning in the Tropos Methodology. *Engineering Applications of Artifcial Intelligence*, 18(2):159–171, March 2005.

[10] G. A. Holton. Defining Risk. *Financial Analyst Journal*, 60(6):1925, 2004.

[11] R. R. Lutz and R. M. Woodhouse. Requirements Analysis Using Forward and Backward Search. *Annals Software Engineering*, 3:459–475, 1997.

[12] NASA. Probabilistic Risk Assessment Procedures Guide for NASA Managers and Practitioners. http://www.hq.nasa.gov/office/codeq/, August 2002.

[13] C. P. Pfleeger and S. L. Pfleeger. *Security in Computing*. Prentice Hall PTR, 3rd edition, 2002.

[14] G. Reniers, W.Dullaert, and K. Soudan. A domino effect evaluation model. Working papers, University of Antwerp, Faculty of Applied Economics, December 2004.

[15] G. G. Roy and T. L. Woodings. A framework for risk analysis in software engineering. In *APSEC '00: Proceedings of the Seventh Asia-Pacific Software Engineering Conference*, page 441, Washington, DC, USA, 2000. IEEE Computer Society.

[16] A. van Lamsweerde and E. Letier. Handling Obstacles in Goal-Oriented Requirements Engineering. *IEEE Transaction Software Engineering*, 26(10):978–1005, 2000.

[17] A. van Lamsweerde, E. Letier, and R. Darimont. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transaction Software Engineering*, 24(11):908–926, 1998.

[18] W. Vesely, F. Goldberg, N. Roberts, and D. Haasl. *Fault Tree Handbook*. U.S Nuclear Regulatory Commission, 1981.

[19] E. Yu. *Modelling Strategic Relationships for Process Engineering*. PhD thesis, University of Toronto, Department of Computer Science, 1995.