Using Risk Analysis to Evaluate Design Alternatives

Yudistira Asnar, Volha Bryl, Paolo Giorgini

Department of Information and Communication Technology
University of Trento, Italy
{yudis.asnar,volha.bryl,paolo.giorgini}@dit.unitn.it

Abstract. Recently, multi-agent systems have proved to be a suitable approach to the development of real-life information systems. In particular, they are used in the domain of safety critical systems where availability and reliability are crucial. For these systems, the ability to mitigate risk (e.g., failures, exceptional events) is very important. In this paper, we propose to incorporate risk concerns into the process of a multi-agent system design and describe the process of exploring and evaluating design alternatives based on risk-related metrics. We illustrate the proposed approach using an Air Traffic Management case study.

1 Introduction

Multi-Agent Systems (MAS) have recently proved to be a suitable approach for the development of real-life information systems. The characteristics they exhibit (e.g., autonomy and ability to coordinate their activities), are indeed useful for safety critical and responsive systems [1]. This is because their subsystems can work independently and respond to events (e.g., failure, exceptional situation, unexpected traffic, etc.) as quick and correct as possible. For instance, a disaster management involves several stakeholders that work autonomously, cooperatively and responsively in unpredictable environments. In this scenario, agents can be used, for example, to assist stakeholders in managing traffic during the rescue period and then reduce the probability of chaotic situations [2].

In a safety critical system, human lives heavily depend on the availability and reliability of the system [3]. For this reason, countermeasures are introduced to mitigate as much as possible the effects of occurring failures. For instance, OASIS Air Traffic Management system [4], which exploits autonomous and responsive agents, is used to manage airspace and schedule air traffic flow. In this case, a designer ensures that agents perform their tasks properly and do not endanger the aircrafts. OASIS implements monitor components/agents that compare the prediction of aircraft locations (i.e., the results of predictor agents) and the actual aircraft position. In case of a significant discrepancy, the monitor agent notifies the scheduler agent to re-schedule the landing time of related aircraft. The introduction of a monitor agent corresponds to a countermeasure to prevent the risk of a collision. However, since designers can not have a complete knowledge about future events/situations, they are not able to elicit all the necessary countermeasures.

A different approach is adopted in the Autonomous Nano Technology Swarm (ANTS) project [1], where three different types of agents (ruler, messenger and worker) cooperate one another in order to explore asteroids. An important feature of ANTS is that, since agents can be damaged or even destroyed by the asteroid, rulers have the ability to re-organize the remaining messenger and worker agents. Basically, this run-time re-organization corresponds to a countermeasure that is adopted to compensate the loss of damaged agents. This introduces at design-time the problem of enabling agents with automatic adaptation capabilities [5] to deal with the effect of failures occurring at run-time.

In [6, 7], we have proposed an approach to support the design of secure and co-operative systems. The main idea was to use planning techniques to find and evaluate possible design alternatives. The objective of this paper is to extend the approach to MAS design and introducing a suitable risk-based metric for evaluating alternatives. We introduce a process based on the following steps:

– system actors, their goals and capabilities, goal decompositions, and possible dependency relationships among actors are identified;
– the above information are passed as input to a planner that search for a possible plan able to satisfy all actors' goals;
– the plan is evaluated w.r.t. risk, that is it is checked whether the risk associated to goals is under a predefined threshold;
– if the evaluation reveals that changes are still necessary, the problem is refined, a new plan is generated and then evaluated.

In safety critical systems, it is important to have a responsible for any decision taken. This requires that the human designer being part of the decisional process and, particularly, being the responsible of the approval of the final solution. Our framework is meant to be a Computer-Aided Software Engineering (CASE) tool that helps a designer in defining and evaluating each design alternative with respect to the associate risk level. The approach can also be used to assist the designer in performing the runtime design of a MAS.

The paper is structured as follows. We start by introducing a case study which then will be used to illustrate our approach. The approach itself is detailed in Section 3, where we explain how the problem of selecting a suitable MAS design can be framed as a planning one, and then in Section 4, where the process of the risk-based evaluation of the obtained alternative design is explained. The application of our approach to the case study is presented in Section 5, which is followed by a short overview of the related work and some conclusive remarks in Sections 6 and 7, respectively.

2 Case Study

In this paper, we use the Air Traffic Management (ATM) case study that is used in the SERENITY Project[1] to validate security and dependability patterns. An ATM is categorized as a safety-critical system because it is closely related to the safety of human lives. Therefore, an ATM system is required to be available and reliable all the time of its operation. However, having a 100% available and reliable system is hardly possible,

[1] http://www.serenity-project.org

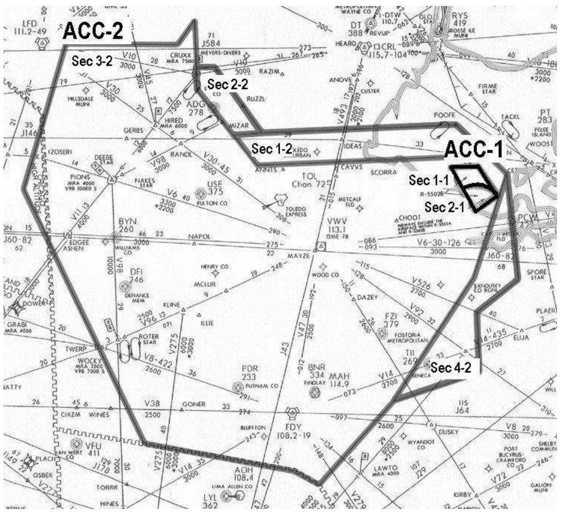


Fig. 1. Airspace Division between ACC-1 and ACC-2

because there are many events that can obstruct the system which can not be known in advance (i.e., during the system development phase). For example, in a specific sector, aircraft traffic can exceed the safety threshold which was not anticipated during the design of the ATM. These events can compromise the availability and reliability of sub-components of the ATM system (e.g., radar processor, CWP[2]).

An Air traffic Control Center (ACC) is a body authorized to provide air traffic control (ATC) services in certain airspace. These services comprise controlling aircraft, managing airspace, managing flight data of controlled aircraft, and providing information about the situation of the air. Suppose there are two adjacent ACCs, namely ACC-1 and ACC-2 (Fig. 1), where the airspace of ACC-1 is surrounded by the airspace of ACC-2. The airspace is organized into several adjacent volumes, called sectors. For instance, the airspace of ACC-1 is divided into sectors (Sec 1-1 and Sec 2-1), and ACC-2 has its airspace divided into 4 sectors (Sec 1-2, 2-2, 3-2, and 4-2). Each sector is operated by a team, consisting of a controller (e.g., Sec 1-1 has C1-1 as a controller), and a planner (e.g., P1-1 is a planner for Sec 1-1). For the sake of communication, several adjacent sectors in an ACC are supervised by a supervisor (e.g., SU1-1 supervises Sec 1-1 and 2-1 and SU1-2 supervises Sec 1-2 and 2-2). In this scenario, the supervisor role is assigned to a human agent, while software agents cover the role of controller and planner. To simplify, we simple call actor, both human agent and software agent. The supervisor also acts as a designer and so, responsibly, approve/decline the new plans. The scenario starts from the normal operation of ATM in which SU1-1 delegates the control of sector 1-1 to team 1 formed by controller C1-1 and planner P1-1.

[2] Controller Working Position (CWP) is a set of resources allocated to support a controller to perform his/her tasks

C1-1 and P1-1 work together providing ATC services to the aircraft in sector 1-1. C1-1 controls aircraft to guarantee the safe vertical and horizontal separation of each aircraft, while P1-1 manages the flight data of the controlled aircraft and the airspace of sector 1-1.

One day during summer holidays, a flight bulletin reports that there will be an increase of the en-route traffic in sector 1-1. According to the analysis made by P1-1, this goes beyond the capacity of a single controller (C1-1). Thus, SU1-1 needs to re-design his sectors in a way that the en-route traffic can be handled safely. He can

- divide the airspace into smaller sectors s.t. each controller covers a smaller area and consequently, the number of sectors that are supervised by SU1-1 is increased; or
- delegate a part of the airspace to the adjacent supervisor (it could be from the same or different ACC).

Each alternative introduces different requirements. For instance, when dividing the airspace, SU1-1 needs to ensure the availability of a controlling team (G_{14}, G_{21} in Table 1) and the availability of a set of CWP (G_{15}, G_{22} in Table 1). Conversely, if SU1-1 decides to delegate a part of his airspace to another supervisor, then SU1-1 needs to define delegation schema (G_{10} in Table 1) and to have sufficient level of "trust" towards the target supervisor and his team to manage the delegated airspace. Moreover, SU1-1 needs to be sure that the target supervisor has sufficient infrastructure (e.g., radar, radio communication coverage) to provide ATC services in the delegated airspace.

The details of the ATM case study are presented in Section 5, including organizational setting and capabilities of each actor. In the following sections, we explain how to encode the case study as a planning problem, and then how to evaluate and refine the candidate plan so to maintain the level of risk below a predefined threshold.

3 Planning Domain

Generating design alternatives can be framed as a planning problem: generating a design alternative means constructing a plan that satisfies the system's goals. The basic idea behind the planning approach is to automatically determine the course of actions (i.e. a plan) needed to achieve a certain goal, where an action is a transition rule from one state of the system to another [8, 9]. Actions are described in terms of preconditions and effects: if a precondition is true in the current state of the system, then the action is performed. As a consequence of an action, the system will be in a new state where the effect of the action is true.

Thus, to define the planning problem, we need to formalize

- the initial and the desired states of the system;
- the actions of the planning domain;
- the planning domain axioms.

In order to represent the initial state of the system (i.e. actor and goal properties, and social relations among actors), first order logic is used with conjunctions of predicates and their negations, specifying the states of the system. To describe our domain we use the following predicates.

- For the goal properties:
  - satisfied(G – goal), which becomes true when the goal G is fulfilled. The predicate is used to define the goal of the planning problem (i.e., to specify, which goals should be satisfied in the final state of the system);
  - and/or_subgoal$_n$(G, G$_1$, G$_2$, ..., G$_n$ – goal) represents the predefined way of goal refinement, namely, it states that G can be and/or-decomposed into n and/or-subgoals;
  - type(G – goal, GT – goal_type) is used to typify goals;
  - criticality_h/m/l(G – goal) represents the criticality of the goal, one of high, medium, or low. The criticality level implies the minimum needed level of trust between the actors when the goal is delegated. For instance, if the criticality of the goal G is high, then it could be delegated from the actor A$_1$ to the actor A$_2$ only if A$_1$ can depend on A$_2$ for the type of goals which G belongs to with the high level of trust.
- For the actor properties:
  - wants(A – actor, G – goal) represents the initial actor's desires;
  - can_satisfy(A – actor, G – goal) and can_satisfy_gt(A – actor, GT – goal_type) are used to represent the capabilities of an actor to satisfy a goal, or a specific type of goal, respectively.
- For the actor dependencies:
  - can_depend_on_gt_h/m/l(A$_1$, A$_2$ – actor, GT – goal_type) means that actor A$_1$ can delegate the fulfillment of the goal of type GT to actor A$_2$, and the trust level of the dependency between these actors for this specific goal type is high, medium, or low, respectively.

A plan, constructed to fulfill the goals, can contain the following actions, defined in terms of preconditions and effects, expressed with the help of the above predicates.

- *Goal satisfaction.* An actor satisfies a goal if it is among its desires (either initially, or after the delegation from another actor), and it has the capability to satisfy it.
- *Goal decomposition.* A goal could be decomposed either into the and-subgoals, meaning that all of them should be satisfied to satisfy the initial goal, or into the or-subgoals, which represent alternative ways of achieving the goal.
- *Goal delegation.* An actor might not have enough capabilities to achieve its goals by itself and therefore, it has to delegate the responsibility of their satisfaction to other actors. As was mentioned before, the delegation can only take place if the level of trust between the actors is not lower than the criticality level required for the goal to be delegated.
- *Goal relaxation.* If there is no way to find a dependency relation which satisfies the required level of trust, then the goal criticality might be relaxed (i.e., lowered). This can be a risky action, as in many cases it is not safe to lower the level of criticality. Therefore, to minimize the risk, as soon as the delegation has been performed, the goal criticality is restored to the original value.

To complete the planning domain, we use axioms which hold in every state of the system and are used to complete the description of the current state. For example, to propagate goal properties through goal refinement, the following axiom is used: a goal is satisfied if all its and-subgoals or at least one of the or-subgoals are satisfied.

We have chosen LPG-td [10], a fully automated system for solving planning problems, for implementing our planning domain. LPG-td supports the PDDL (Planning Domain Definition Language) 2.2 specification, which was used to formalize system states, actions and axioms described above. The details on how and why this planner has been chosen have been addressed in [6]. We also refer the reader to [6] and [7] for the details on how the actions and axioms of the planning domain were implemented in PDDL 2.2.

4 Evaluation Process

After a design alternative, called a candidate plan, is generated by the planner, it should also be evaluated and modified based on a number of criteria, and finally approved by a designer. By modifying the candidate plan we mean refining the problem definition by identifying the actions that should be avoided to get the less risky design alternative. The refinement of the problem definition is followed by replanning.

Previously, we proposed a way of evaluating a candidate plan, which is based on the load distribution concerns [7]. It is assumed that the actors want to keep the number and complexity of actions they are involved in, below the predefined thresholds. In this work, we propose another form of evaluation, namely adopting a risk evaluation metric. The goal of the iterative planning-and-evaluation procedure is to select a plan among the available alternatives that has an acceptable level of risk. In this framework, we consider two types of risk. The first type is the risk about the satisfaction of a goal, called satisfaction risk (sat-risk). Sat-risk represents the risk of a goal being denied/failed when an actor attempts to fulfill it. The value of this risk is represented in terms of the following predicates: *FD* (Fully Denied), *PD* (Partially Denied), and *ND* (Not Denied). These predicates are taken from [11], and represent the high, medium, and low level of sat-risk, respectively. The second type of risk is related to the risk of goal delegation. It is based on the requirement that the level of trust between two actors should match the criticality of the delegated goal. For instance, if a link between two agents is highly trusted, than it can be used for delegating goals of any criticality level, but if the level of trust of a delegation link is medium then only goals with low and medium criticality can be delegated through this link, and the risk is introduced when the criticality of a goal should be lowered before it could be delegated.

The process of selecting a suitable design alternative is illustrated in Algorithm 1, which should be run twice. In the first execution, the algorithm constructs a plan without any relaxation actions (i.e., *relax*=false). If there is no solution then the second execution is preformed allowing relaxation actions. Some steps in the algorithm are fully automated (e.g., *run_planner* line 3), while some still need a human involvement (e.g., adding the allowed actions to the *whitelist* in line 7). The algorithm is iterative and comprises the following phases: planning, evaluation, and, finally, plan refinement. There are two evaluation steps in the algorithm: STEP-1 evaluates the risks of goal satisfactions (line 4), and STEP-2 evaluates relaxation actions (line 6). The first execution does only STEP-1, and if the second execution is necessary, both STEP-1 and STEP-2 are

Algorithm 1 Planning and Evaluation Process

```
Require: domain {domain description in PDDL}
    problem {goal and initial state of the problem in PDDL}
    whitelist {a list of allowed action}
    relax{allow/not relaxation}
 1: boolean finish←false
 2: while not finish do
 3:     plan ←$run\_planner(domain, problem, relax)$
 4:     if not $evaluate\_sat(plan)$ then
 5:         $refine\_sat(plan, problem)$
 6:     else if relax and not $evaluate\_act(plan)$ then
 7:         $refine\_act(plan, problem, whitelist)$
 8:     else
 9:         finish ←true
10:     end if
11: end while
12: return plan
```

executed. Each evaluation step is followed by a refinement action (line 5 or 7), which aims at changing the planner input s.t. during the next iteration it will produce the better (i.e. less risky) candidate plan. In the following we give details on the two evaluation steps of the algorithm.

STEP 1: Goal Satisfaction Evaluation

After a candidate plan is elicited (line 3), it should be evaluated and refined, s.t. it meets the requirements imposed on it (i.e., the level of risk associated with the plan is below the predefined threshold). The aim of the first evaluation step (line 4 of the Algorithm) is to assure that sat-risk values of the candidate plan, i.e. the likelihood of each system goal being denied/failed, are at most equal to the accepted ones, specified by a designer.

By examining the candidate plan, the goal model of each top goal can be constructed, as the one in Fig. 3. A goal model shows how a top goal is refined into atomic tangible leaf goals, i.e. for each leaf goal there is an actor that can fulfill it. Starting from the sat-risk values of leaf goals, the risk values are propagated up to the top goals with the help of so called forward reasoning. Forward reasoning is an automatic reasoning technique introduced in [11], which takes a set of sat-risk of leaf goals as an input. Notice that sat-risk value depends on which actor satisfies the leaf goal according to the candidate plan. The algorithm propagates the qualitative values assigned to the leaf goals along the goal tree up to the top goal, and thus the corresponding value for the top goal is calculated.

If sat-risk of one top goal is higher than the specified threshold, then the refinement process needs to be performed. The refinement (line 5) identifies those assignments of the leaf goals to actors that should be avoided in order to have the sat-risk values of the top goals within the specified thresholds. The refinement process starts by generating a possible set of assignment (i.e., sat-risk values of the leaf goals) that results in the top goals having the sat-risks below the specified thresholds. This set of assignments is called a reference model. Basically, the reference model is a set of maximum sat-risk values of leaf goals that results in the top goals, which sat-risks do not violate the thresholds. If the sat-risk values of leaf goals in the goal model are below the maximum specified in the reference model, then the sat-risk of the top goals are acceptable. The reference model can be obtained automatically using backward reasoning [12], which aims at constructing the assignments of leaf goals to actors s.t. the specified sat-risk value for the top goals are achieved. According to [12], a goal model is encoded as a satisfiability (SAT) problem, and a SAT solver is used to find the possible assignments that satisfy the SAT formula.

By comparing the sat-risk values of leaf goals in the goal model with the corresponding values in the reference model, the riskier goal satisfaction actions are detected (i.e., the leaf goal in the goal model that has higher sat-risk than the corresponding value in the reference model). For instance, in Fig. 3 the risk-sat of goal capable managing airspace (G_{19}) that is satisfied by actor P1-1 is FD (Fully Denied), while according to the reference model, the value of G_{19} should be at most PD (partially denied). Therefore, the problem definition needs to be refined s.t. P1-1 does not satisfy G_{19}. However, we can not refine the problem by simply specifying G_{19} must not be satisfied by P1-1, because in Fig. 3 the goal model states that the satisfaction of G_{19} by P1-1 is too risky. Ideally, we specify "the path of actions" from the top goal that lead to the goal G_{19} being satisfied by P1-1. To simplify the refinement process, we only consider one action involving G_{19}, performed just before P1-1 satisfies it. In case of Fig. 2(b), such an action (called *previous related action*) is $and\_decompose$ G_3 into G_{18} and G_{19}, which is performed by P1-1. Thus, the refinement predicate that should be introduced in the problem definition is the following.

$$\neg(satisfy(P_{1-1}, G_{19}) \wedge and\_decompose_2(P_{1-1}, G_3, G_{18}, G_{19}))$$

After the problem definition is refined, the planner is run again to elicit a new candidate plan using the refined problem definition. All the above described steps can be done automatically, without any interference a designer.

STEP 2: Action Evaluation

The second evaluation step (line 6 of the Algorithm) is performed to guarantee that the relaxation actions in a candidate plan are acceptable/not risky. In our framework, we assume that relaxing the criticality of a goal from high to medium, or from medium to low, can be performed safely only by the owner of a goal. We say that goal G is owned by A if G was initially wanted by A (i.e., in the initial state A wants G to be satisfied). In this case all the subgoals of G are also said to be owned by A. We use the term *further relaxation* to refer to the situation when the relaxation is done by another actor (i.e., not the owner). *Further relaxation* is assumed to be a risky action, but sometimes it is impossible to find a plan without it. This action could be allowed by the interference of a designer adding it to the *whitelist*.

For instance, in the ATM case study SU1-1 intends to increase his airspace capacity in response to the traffic increase by delegating his airspace (G_{11}) to SU1-2. As the fulfillment of G_{11} is critical (the criticality level is high), SU1-1 needs to have high trust level towards SU1-2 for delegating G_{11} (i.e., $can\_depend\_on\_gt\_h(SU_{1-1}, SU_{1-2}, G_{11})$ should be true). Later, SU1-2 refines G_{11} into the subgoals control the aircraft (G_2) and manage the airspace (G_3). For satisfying these goals, SU1-2 needs to depend on the controller C1-2 for G_2, and on the planner P1-2 for G_3, because they are the ones that have the capabilities to satisfy the corresponding goals. Let us assume the trust level of the dependency of SU1-2 towards C1-2 for G_2 is medium. Thus, SU1-2 needs to *further relax* the criticality of G_2 s.t. it can be delegated to C1-2.

Basically, the evaluation aims to guarantee that there is no relaxation action taken by an actor which is not the owner of the goal. Otherwise, the designer needs to explicitly allow the actor to do this action (i.e., add it to the *whitelist*). Notice that relaxation actions are introduced only in the second run of algorithm. During the refinement phase (line 7) the problem definition is changed to meet this requirement, which is followed by replanning.

5 Experimental Results

In this section, we illustrate the application of our approach to the ATM case study. The following subsections detail the case study formalization, and the planning-and-evaluation process, performed in accordance with Algorithm 1. The aim of the process is to elicit an appropriate plan for SU1-1's sector, taking into account the constraints and the risk of each alternative. The scenario starts with the intention of SU1-1 to increase the capacity of airspace (G_6) as a response to the air traffic increase in sector 1-1. SU1-1 faces a problem that C1-1 is not available to control (G_{14}) more traffic. Therefore, SU1-1 needs to modify sector 1-1 without involving C1-1 s.t. the increase of air traffic can be handled.

5.1 Case Study Formalization

The following inputs should be provided for Algorithm 1:

– A formalized problem definition, which contains all the properties of the actors of the ATM system, and their goals. The complete list of properties can be found in Table 2.
– Goals of the planning problem (e.g., satisfy G_6 without involving C1-1 in satisfying G_{14}).
– A list of authorized further relaxation actions ($whitelist$).
– Risk values of goal satisfaction actions. Table 1 shows all sat-risk values of the satisfaction actions.
– Accepted risk values (e.g., risk value of G_6 is at most PD).

In Table 1, the goal criticality values are presented in column Crit. Goal criticality (high, medium, or low) denotes a minimum level of trust between two actors that is needed if one actor decides to delegate the goal to another actor. For instance, goal manage airspace (G_3) is categorized as a *highly critical* goal, and goal analyze air

Actor / Goal			C1-1	C2-1	P1-1	P2-1	SU1-1	C1-2	P1-2	SU1-2
Id.	Description	Crit.								
G1	Manage Aircraft within ACC									
G2	Control Aircraft	H								
G3	Manage Airspace	H								
G4	Manage Flight Data	M						PD		
G5	Maintain Air Traffic Flow in Peak-Time									
G6	Increase Airspace Capacity									
G7	Analyze Air Traffic	L								
G8	Re-sectorize within ACC									
G9	Delegate Part of Sector									
G10	Define Schema Delegation	M			ND					PD
G11	Delegate Airspace	H								
G12	Have Controlling Resources									
G13	Have Capability to Control the Aircraft		ND	PD				PD		
G14	Avail to Control		FD	ND						
G15	Have Control Working Position for Controller	H					ND			PD
G16	Have Authorization for FD Modification	M	ND	ND						
G17	Have Capability to Manage FD				ND	PD				
G18	Have Resources for Planning	M					ND			
G19	Have Capability to Manage Airspace				FD	PD			PD	
G20	Have Capability to Analyze Air Traffic				PD					
G21	Avail to Plan				ND	ND			ND	
G22	Have Control Working Position for Planner	H					ND			ND

Table 1. Goals Criticality and Satisfaction Risk (Criticality = H: High, M: Medium, L: Low and sat-risk = Full Denied, Partial Denied, and Not Denied)

traffic(G_8) has *low* criticality. Thus, these goals require different level of trust for being delegated to another actor.

Moreover, Table 1 shows the risk levels of satisfying a goal when an actor tries to achieve it. Note that, the sat-risk level of a goal depends on which actor satisfies the goal. sat-risk takes one of the tree values: *FD* (Fully Denied), *PD* (Partially Denied), or *ND* (Not Denied). For instance, the table states G_{19} can be satisfied either by actor P1-1, P2-1, or P1-2, and each actor has different level of risk (sat-risk) – *full*, *partial*, and *partial*, respectively. The empty cells in Table 1 imply the actor does not have capabilities to fulfill the corresponding goal.

Table 2 shows properties of actors and their goals in ATM case study. Namely, it represents actor capabilities (can_satisfy), possible ways of goal refinements (decompose), and possible dependencies among actors (can_depend_on) together with the level of trust for each dependency. For instance, actor SU1-1 can_satisfy goals G_{15}, G_{18}, and G_{22}, and the actor has knowledge to decompose G_1, G_5, G_6, G_8, and G_9. And SU1-1 has *high* level of trust to delegate G_2 to C1-1 or C2-2. The same intuition is applied for the other cells.

5.2 Planning and Evaluation Process

STEP 0: Planning. After specifying the inputs, the planner is executed to elicit a candidate plan to fulfill the predefined goals, which is shown in Fig. 2(a). These goals state that the plan should satisfy G_6, and the solution should not involve C1-1 to satisfy G_{14} because C1-1 is already overloaded controlling the current traffic. Moreover, the plan-

Actor	can_satisfy	decompose			can_depend_on		
		type	top-goal	sub-goals	level	dependum	dependee
SU1-1	G15	And	G1	G2, G3	H	G2	C1-1, C2-1
	G18	And	G5	G6, G7	H	G3	P1-1, P2-1
	G22	Or	G6	G8, G9	H	G4	P1-1, P2-1
		And	G8	G2, G3, G4	M	G7	P1-1
		And	G9	G10, G11	M	G10	P1-1
					L	G10	SU1-2
					H	G11	SU1-2
P1-1, P2-1	G17	And	G3	G18, G19	H	G22	SU1-1
	G19	And	G4	G16, G17, G18			
	G21	And	G7	G18, G20			
P1-1	G10				L	G16	C1-1
	G20						
P1-2					L	G16	C2-1
C1-1, C2-1	G13	And	G2	G4, G12, G13	H	G15	SU1-1
	G14	And	G12	G14, G15			
	G16						
C1-1					M	G4	P1-1
C2-1					M	G4	P2-1
					M	G4	P1-1
SU1-2	G10	And	G11	G2, G3	M	G2	C1-2
	G15				M	G3	P1-2
	G22						
P1-2	G19	And	G3	G19, G21, G22	M	G22	SU1-2
	G21						
C1-2	G4	And	G2	G4, G13, G15	M	G15	SU1-2
	G13						

Table 2. List of Actors and Goal Properties for the ATM Case Study. (Level of trust: H: High, M: Medium, L: Low)

ner should not involve the other ACC (i.e., SU1-2) by avoiding the delegation of G_{11} to SU1-2 even it is possible in Table 2. Before adopting the candidate plan (Fig. 2(b)), two evaluation steps explained in previous section should be performed to ensure the risk of the candidate plan is acceptable.

STEP 1: Goal Satisfaction Evaluation assesses the satisfaction risk of a candidate plan. The goal model of goal G_6 (in Fig. 3) is constructed on the basis of the candidate plan (in Fig. 2(b)). The goal model shows which actors are responsible for satisfying the leaf goals. For instance, G_{19} must be satisfied by P1-1 and, moreover, in this scenario, G_9 is left unsatisfied because the other or-subgoal, G_8, was selected to satisfy G_6.

In this scenario, we assume that the acceptable sat-risk value for G_6 is PD. To calculate the sat-risk value of goal G_6, forward reasoning is performed (i.e., the sat-risk values of leaf goals in Table 1 are propagated up to the top goal). This reasoning mechanism is a part of the functionality of the GR-Tool[3], a supporting tool for goal analysis. By means of the forward reasoning, we obtain that the sat-risk of G_6 is FD, which is higher than the acceptable risk (i.e., PD). Thus, the refinement is needed to adjust the problem definition, so that a less risky plan is constructed during the next replanning. The refinement starts with the elicitation of a reference model using backward reasoning. The reference model specifies that all leaf goals must have at most PD sat-risk value in order the sat-risk of top goal G_6 not to be higher than PD.

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

```
(satisfied G6)
(not(satisfy C1-1 G14))
(not(delegate SU1-1 SU1-2 G11))
```

(a) Goal of Problem Definition

```
0: (OR_DECOMPOSES2 SU1-1 G6 G8 G9)
1: (AND_DECOMPOSES3 SU1-1 G8 G2 G3 G4)
2: (DELEGATES SU1-1 C2-1 G2)
3: (AND_DECOMPOSES3 C2-1 G2 G4 G12 G13)
4: (SATISFIES C2-1 G13)
5: (AND_DECOMPOSES2 C2-1 G12 G14 G15)
6: (SATISFIES C2-1 G14)
7: (DELEGATES C2-1 SU1-1 G15)
8: (SATISFIES SU1-1 G15)
9: (DELEGATES C2-1 P2-1 G4)
10: (DELEGATES SU1-1 P2-1 G4)
11: (AND_DECOMPOSES3 P2-1 G4 G16 G17 G18)
12: (SATISFIES P2-1 G17)
13: (DELEGATES P2-1 SU1-1 G18)
14: (SATISFIES SU1-1 G18)
15: (RELAX2L P2-1 G16)
16: (DELEGATES P2-1 C2-1 G16)
17: (SATISFIES C2-1 G16)
18: (DELEGATES SU1-1 P1-1 G3)
19: (AND_DECOMPOSES2 P1-1 G3 G18 G19)
20: (SATISFIES P1-1 G19)
```

(b) The Candidate Plan after STEP 0

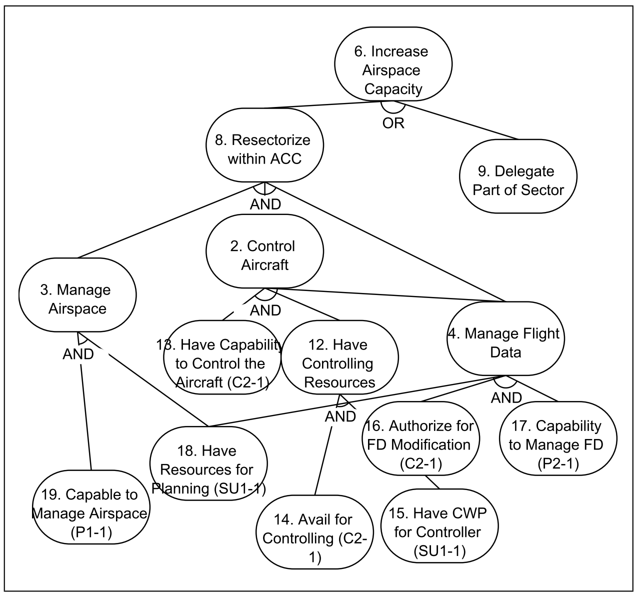
Fig. 2. Plan for Increasing Air Space Capacity

By comparing the sat-risks of leaf goals in the goal model with the reference model, G_{19} (satisfied by P1-1) is detected to be a risky goal; its sat-risk (in Table 1) is FD which is higher than the one in the reference model. Therefore, the problem definition is refined to avoid P1-1 satisfying G_{19}. As G_{19} is a subgoal of G_3, the decomposition action is also negated, as the previous related action, according to the procedure explained in the previous section. Thus, the problem definition is refined, and the goal of the planning problem is now of the form shown in Fig. 4(a). Afterwards, the planner is run to elicit a new candidate plan. Basically, the new candidate plan is almost the same with the previous plan (Fig. 2(b)), the only difference is in lines 18-20 (see Fig. 4(b)). Later, this candidate plan is evaluated by going through the next step to ensure all the actions (especially, further relaxations) are acceptable in terms of risks.

STEP 2: Action Evaluation filters the malicious relaxation actions. The scenario starts from the goal G_6 which is wanted by SU1-1. As all the other goals of the candidate plan are the result of the refinement of G_6, the owner of all of them is again SU1-1. Thus, relaxing the criticality of any goals that is performed by any actors except SU1-1 is seen as a risky action.

For instance, P2-1 relaxes the criticality of G_{16} (line 15 in Fig. 2(b)) to low instead of medium. By default this action is a risky one and should be avoided, unless the designer states explicitly that this action is not risky by adding it to the *whitelist*. Once it is considered unacceptable, the goal of the planning problem should be extended with the negation of the relaxation action (i.e., (not (relax2l P2-1 G1))).

Moreover, the designer can also introduce rules to avoid certain actions. For instance, the designer may prevent C2-1 from delegating G_4 to P2-1 (line 9 in



6. Increase Airspace Capacity
OR
8. Resectorize within ACC
9. Delegate Part of Sector
AND
2. Control Aircraft
3. Manage Airspace
AND
4. Manage Flight Data
13. Have Capability to Control the Aircraft (C2-1)
12. Have Controlling Resources
AND
AND
AND
16. Authorize for FD Modification (C2-1)
17. Capability to Manage FD (P2-1)
18. Have Resources for Planning (SU1-1)
19. Capable to Manage Airspace (P1-1)
14. Avail for Controlling (C2-1)
15. Have CWP for Controller (SU1-1)

Fig. 3. The Goal Model of Candidate Plan in Fig. 2(b)

```
(satisfied G6)
(not (satisfy C1-1 G14))
(not (delegate SU1-1 SU1-2 G11))
(not (and (satisfy P1-1 G19)(and_decompose2 P1-1 G3 G18 G19)))
```

(a) Problem Definition Refinement after STEP 1

```
............
18: (DELEGATES SU1-1 P2-1 G3)
19: (AND_DECOMPOSES2 P2-1 G3 G18 G19)
20: (SATISFIES P2-1 G19)
```

(b) Final Plan for Satisfying G_6

Fig. 4. Final Problem Definition and Plan for increase the airspace capacity (G_6)

Fig. 2(b)) by adding a new predicate to the goal of the planning problem (namely, (not (delegate C2-1 P2-1 G4))). For the sake of simplicity all the possible relaxation actions in the candidate plan are put to the *whitelist*, so we do not refine the problem definition any further.

Thus, the last candidate plan to redesign SU1-1's sector is approved s.t. the traffic increase can be handled. Moreover, the plan is guaranteed to have risk values less/equal than the predefined thresholds (i.e., sat-risk of G_6 is less or equal than PD).

6 Related Work

Several approaches have been proposed in literature to use risk analysis in the design of a software system. CORAS [13] has been developed as a framework for risk analysis of security critical systems. Basically, CORAS consists of context identification, risk identification, risk analysis, risk evaluation, and risk treatment. CORAS can be integrated with other risk modeling frameworks, such as Failure Mode, Effects, and Criticality Analysis (FMECA) [14], Fault Tree Analysis (FTA) [15], Hazard and Operability (HAZOP) [16]. This methodology has been tested with security systems, especially in the E-Commerce and Telemedicine area. In reliability engineering community, Defect Detection and Prevention (DDP) [17] has been proposed to assess the impact of risk and related mitigation to the system. The DDP framework deals with three types of data: Objective, Risk, and Mitigation. The objective is defined as the goal the system has to achieve. The risk is defined as the thing that, once it occurs, leads to the failure of the objective. Finally, the mitigation is a course of actions that can be applied to reduce the risk. With the help of DDP the designer can assess how the introduction of a mitigation impacts to the objectives. In this approach, a designer must construct the system design before assessing the risk. Our approach, on the other hand, is aimed to automate both the design and its evaluation.

The field of AI planning has been intensively developing during the last decades, and has found a number of interesting applications (e.g., robotics, process planning, autonomous agents, etc.). There are two basic approaches to the solution of planning problems [8]. One is graph-based planning algorithms in which a compact structure, called Planning Graph, is constructed and analyzed. In the other approach the planning problem is transformed into a SAT problem and a SAT solver is used. There exist several ways to represent the elements of a classical planning problem (i.e. the initial state of the world, the system goal, or the desired state of the world, and the possible actions system actors can perform). The most widely used, and to a certain extent standard representation is PDDL (Planning Domain Definition Language), the problem specification language proposed in [18]. Current PDDL version, PDDL 2.2 [19] used during the last International Planning Competition [20], supports many useful features (e.g., derived predicates and timed initial literals).

A few works can be found which relate planning techniques with information system design. In [21] a program called ASAP (Automated Specifier And Planner) is described, which automates a part of the domain-specific software specification process. ASAP assists the designer in selecting methods for achieving user goals, discovering plans that result in undesirable outcomes, and finding methods for preventing such outcomes. The disadvantage of the approach is that the designer still performs a lot of work manually while determining the combination of goals and prohibited situations appropriate for the given application, defining possible start-up conditions and providing many other domain-specific expert knowledge. Some works present a planning application to assist an expert in designing control programs in the field of Automated Manufacturing [22]. The system they have built integrates POCL (Partial Order Causal Link), hierarchical and conditional planning techniques [22, 9]. The authors consider standard planning approaches to be not appropriate with no ready-to-use tools for the real world, while in our paper the opposite point of view is advocated, and the off-the-shelf planner is used.

7 Conclusion

In this paper, we have proposed an approach to incorporate risk analysis into the process of MASs design. The approach is based on the use of planning to explore the space of alternative designs and risk-based evaluation metrics to evaluate the resulting solutions. We argue that the approach is particularly suitable for the design of critical and responsive systems, such as air traffic management, health-care systems, disaster management (e.g., post-disaster urban planning), traffic management, etc.

The proposed framework is meant to support a designer in generating, exploring, and evaluating design alternatives either during the initial, classical design, or during runtime redesign of a MAS. We consider runtime redesign of high importance for modern information systems, which operates in continuously changing environment and then require highly adaptable characteristics. Among the limitations of our approach, we would like to mention that it only supports a centralized viewpoint (i.e., the designers viewpoint), while the different actors of a system may have different priorities and criticalities. We consider this issue being an interesting direction for future work.

8 Acknowledgments

We would like to thank to Gabriel Kuper, Sameh Abdel-Naby, Hamza Hydri Syed, and anonymous reviewers for all the useful comments. This work has been partially funded by EU Commission, through the SENSORIA and SERENITY projects, by the FIRB program of MIUR under the ASTRO project, and also by the Provincial Authority of Trentino, through the MOSTRO project.

References

1. Truszkowski, W., Rash, J., Rouff, C., Hinchey, M.: Asteroid exploration with autonomic systems. In: Engineering of Computer-Based Systems, 2004. Proceedings. 11th IEEE International Conference and Workshop on the. (May 2004) 484–489
2. Matsui, H., Izumi, K., Noda, I.: Soft-restriction approach for traffic management under disaster rescue situations. In: ATDM'06: 1st Workshop on Agent Technology for Disaster Management. (2006)
3. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.E.: Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Trans. Dependable Sec. Comput. **1**(1) (2004) 11–33
4. Ljungberg, M., Lucas, A.: The OASIS Air-Traffic Management System. In: PRICAI'92: In Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence. (1992)
5. Truszkowski, W., Hinchey, M., Rash, J., Rouff, C.: Autonomous and autonomic systems: a paradigm for future space exploration missions. Systems, Man and Cybernetics, Part C, IEEE Transactions on **36**(3) (2006) 279–291
6. Bryl, V., Massacci, F., Mylopoulos, J., Zannone, N.: Designing security requirements models through planning. In: CAiSE'06, Springer (2006) 33–47
7. Bryl, V., Giorgini, P., Mylopoulos, J.: Designing cooperative IS: Exploring and evaluating alternatives. In: CoopIS'06. (2006) 533–550

8. Weld, D.S.: Recent Advances in AI Planning. AI Magazine **20**(2) (1999) 93–123
9. Peer, J.: Web Service Composition as AI Planning – a Survey. Technical report, University of St. Gallen (2005)
10. LPG Homepage: LPG-td Planner. http://zeus.ing.unibs.it/lpg/
11. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Formal Reasoning Techniques for Goal Models. Journal of Data Semantics (October 2003)
12. Sebastiani, R., Giorgini, P., Mylopoulos, J.: Simple and Minimum-Cost Satisfiability for Goal Models. In: CAISE '04: In Proceedings International Conference on Advanced Information Systems Engineering. Volume 3084., Springer (June 2004) 20–33
13. Fredriksen, R., Kristiansen, M., Gran, B.A., Stolen, K., Opperud, T.A., Dimitrakos, T.: The CORAS framework for a model-based risk management process. In: Safecomp '02: In Proceedings Computer Safety, Reliability and Security. Volume LNCS 2434., Springer (2002) 94–105
14. DoD: Military Standard, Procedures for Performing a Failure Mode, Effects, and Critical Analysis (MIL-STD-1692A). U.S. Department of Defense (1980)
15. Vesely, W., Goldberg, F., Roberts, N., Haasl, D.: Fault Tree Handbook. U.S Nuclear Regulatory Commission (1981)
16. USCG: Risk Based Decision Making Guidelines. http://www.uscg.mil/hq/g-m/risk/e-guidelines/RBDMGuide.htm (November 2005)
17. Feather, M.S.: 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, IEEE Computer Society (November 2004) 391–402
18. Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL – The Planning Domain Definition Language. In: Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems. (1998)
19. Edelkamp, S., Hoffmann, J.: PDDL2.2: The language for the classical part of the 4th international planning competition. Technical Report 195, University of Freiburg (2004)
20. IPC-4 Homepage: International Planning Competition 2004. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/
21. Anderson, J.S., Fickas, S.: A proposed perspective shift: viewing specification design as a planning problem. In: IWSSD '89: 5th Int. workshop on Software specification and design. (1989) 177–184
22. Castillo, L., Fdez-Olivares, J., Gonzlez, A.: Integrating hierarchical and conditional planning techniques into a software design process for automated manufacturing. In: ICAPS 2003, Workshop on Planning under Uncertainty and Incomplete Information. (2003) 28–39