Optimal Hybrid Fault Recovery in a Team of Unmanned Aerial Vehicles [⋆]

M. M. Tousi and K. Khorasani

Department of Electrical and Computer Engineering, Concordia University, Montréal, Québec, Canada H3G 1M8

Contact email: kash@ece.concordia.ca

Abstract

This paper introduces and develops an optimal hybrid fault recovery methodology for a team of unmanned vehicles by taking advantage of the cooperative nature of the team to accomplish the desired mission requirements in presence of faults/failures. The proposed methodology is developed in a hybrid framework that consists of a low-level (an agent level and a team level) and a high-level (discrete-event systems level) fault diagnosis and recovery modules. A high-level fault recovery scheme is proposed within the discrete-event systems (DES) supervisory control framework, whereas it is assumed that a low-level fault recovery designed based on classical control techniques is already available. The low-level recovery module employs information on the detected and estimated fault and modifies the controller parameters to recover the team from the faulty condition. By taking advantage of combinatorial optimization techniques a novel reconfiguration strategy is proposed and developed at the high-level so that the faulty vehicles are recovered with minimum cost to the team. A case study is provided to illustrate and demonstrate the effectiveness of our proposed approach for the icing problem in unmanned aerial vehicles, which is a well-known structural problem in the aircraft industry.

Key words: Hierarchical fault diagnosis; System failure and fault recovery; Autonomous network of unmanned aerial vehicles, Discrete-event systems

1 Introduction

In recent years, there has been a great deal of interest in applications of unmanned aerial vehicles (UAVs) intended for various civilian and military missions [4, 17]. Eliminating the function of a pilot from the aircraft and replacing it with a completely autonomous flight guidance and navigation system complicates and introduces a number of novel challenges [13]. UAVs rely heavily on flight control systems that are expected to be sufficiently robust and intelligent to monitor or even anticipate (prognose) problems with the vehicle flight dynamics. Other challenges such as communication delays might lead to delayed control inputs that could influence the vehicle's stability and control. A UAV reliability study was carried out by the US Department of Defense [13] that shows that the major environmental factors in this regard are precipitation, icing, and the wind. These environmental factors reduce the UAV's reliability and availability for planned missions. A cooperative decision and control theoretic framework is of special interest to mission planners and managers for addressing robust dynamic control of distributed UAVs executing multiple interrelated tasks with a high degree of decentralization [1, 24].

In practical situations and environments, autonomous systems are prone to anomalies, and early detection and diagnosis of faults while the system is operating in a controllable region can help avoid abnormal event progression and reduce performance degradation. There is a large body of literature on fault detection, isolation, and recovery (FDIR) techniques ranging from analytical methods to artificial intelligence and statistical approaches [23]. Anomalies and faults could be present in the actuators, the sensors, or the system components, in general.

On the other hand, discrete-event systems (DES) provides a powerful tool for modeling a wide variety of applications, and has been utilized to address the fault detection and recovery problems [12]. In the supervisory control framework that was proposed by Ramadge and Wonham [26], the DES is modeled as a generator of a formal language, which can be controlled through an external supervisor by enabling or disabling certain events (transitions). This enablement or disablement of events is carried out to restrict the system behavior in order to

[⋆] This research is supported in part by Strategic Projects and Discovery program grants from the Natural Sciences and Engineering Research Council of Canada (NSERC).

satisfy a variety of criteria. Safety specifications, such as the avoidance of prohibited regions of the state-space or the observation of service priorities, could be incorporated among the possible criteria.

Moreover, various optimal control techniques in DES are reported in [5], [8], [20], and [25]. Most works discuss properties of optimal state machines and existence of solutions ([5], [8], and [20]). The method presented in [25], which is also based on other prior work such as [20], proposes to use probabilities within the DES formulation to obtain the optimum solution. On the other hand, [20] introduces a control cost and a path cost, where corresponding to each condition an optimum solution is obtained. Moreover, in [20] the event costs are the same from any state to any others, whereas in [25] the event costs can be different based on the departing and the destination states.

A number of fault recovery (FR) techniques have been proposed in the literature that focus on either high-level (DES supervisory) [19] or simply low-level solutions [2]. On the other hand, not much work has been carried out on the notion of hybrid FR for a team of autonomous unmanned vehicles (UVs). In [21], a novel hybrid FDIR approach for a team of multi-agent systems is proposed. The decision making process is carried out in the high-level DES supervisory unit in a centralized fashion. The main objective of this unit is to obtain the best achievable performance for the team. The low-level FDIR unit, on the other hand, complements the high-level supervisor by monitoring the agents for detection and identification of faults, and by applying the high-level recovery decisions to the agents. One of the distinct features of the proposed hybrid technique is its suitability for accommodating a broad class of faults. However, the proposed method in [21] did not consider an optimum hybrid FDIR solution, and more specifically an optimal hybrid FR solution. The supervisory control in the high-level allows the system to freely go through the feasible solutions considering the given specifications and constraints. This paper generalizes the solution that was provided in [21] by analyzing the available and feasible solutions and determining the optimum paths possible by utilizing combinatorial optimization techniques.

2 Main Results

In this section, our novel approach for tackling the optimal hybrid fault recovery problem in a team of UAVs is formally introduced. This work focuses on the recovery solution at the high-level (DES level). Consequently, it is assumed that the designer has already developed low-level fault diagnosis (FD) as well as low-level fault recovery (FR) modules. Therefore, these issues are beyond the scope of this work and are not described in more detail subsequently.

The multi-agent team framework [21] is used to transform the fault diagnosis and recovery problems into the DES framework. The objective of the high-level fault recovery module is to recover from faults that are not recoverable at the agent level (that is, at the low-level). The high-level recovery module employs an optimal reconfiguration solution which is found by taking advantage of the available combinatorial optimization methods in the DES framework. The schematic of our proposed hybrid fault recovery solution is depicted in Fig. 1.

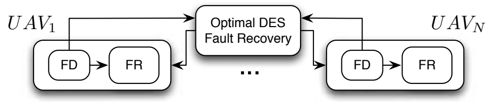

Fig. 1. Our proposed hybrid framework for a team of N UAVs (Legend: FD = fault diagnosis and FR = fault recovery).

The i-th UAV, that is denoted by UAV_i, $i = 1, \ldots, N$, is modeled as a linear time-invariant (LTI) system. It is assumed that there exists a set of controllers that are designed for the UAV_i. In case that the designed controllers for the UAV_i can handle and compensate for all the faulty situations the low-level fault recovery can recover the i-th UAV from the faulty modes. However, when sufficient number of controllers are not available at the low-level FDIR module to compensate for all the faults, the low-level fault recovery solution cannot recover the i-th UAV from all the possible faults that can potentially occur. Therefore, the high-level FR module should either prevent the UAV_i from reaching circumstances where the low-level recovery solution cannot recover the UAV_i from the fault, or that it should minimize the effects of the unrecoverable faults on the other team members by reconfiguring the team members. In the following, we introduce our high-level optimal hybrid fault recovery module which is capable of accomplishing these desired goals and objectives.

2.1 Team of Agents Framework

The team framework employed here is adopted from [21]. In this framework, mission refers to a particular set of tasks that are to be accomplished by the multi-agent system within a given time and a given resource limitation. The main goal of a team of unmanned vehicles is to accomplish a mission in a cooperative manner. Each mission may be represented by and include different team structures, information topologies, and communication architectures.

The above notions are formally defined in [21]. Briefly stated, team structure (TS) in a group of agents refers to the physical location of agents with respect to each other for accomplishing a particular task. Information topology (IT) is defined as the information flow structure which reflects the accessibility and availability of different agents' measurements and sensing for the purpose of computing the control signal in each local controller. The communication architecture (CMA) refers to the configuration of communication links among agents that represents the transfer of information among agents.

Definition 1 *Priority (Pr) is defined as the influence level of each agent on the overall mission performance and is associated with and characterized by the information topology and the communication architecture.*

The notion of priority Pr can be formally quantified as $Pr = IT \times W_{IT} + CMA \times W_{CMA}$, where IT and CMA represent the influence levels associated with the information topology and the communication architecture, respectively. Furthermore, W_{IT} and W_{CMA} are the weights that represent the relative importance of the information topology and the communication architecture on the mission, respectively. For instance, in the leader-follower formation one can envisage two levels of priorities, namely the leader and the followers. One can assign the priority of the leader to be N (the number of team members) times greater than the priority of each follower. Due to communication constraints, each agent is able to communicate with only a limited number of agents. For instance, the influence level of an agent may be decided according to the number of other agents to which it can communicate with.

The first step in constructing a hybrid fault recovery module is to transform the multi-agent team framework introduced above into the DES framework. This process is described in detail in the next subsection.

2.2 DES Framework for Fault Recovery of a Team of UAVs

Let the team model be described in the DES framework by the five-tuple $G = (Q, \Sigma, \delta, q_0, Q_m)$. The tuples are described below in more detail:

- Q denotes the set of team states,
- Σ represents the set $\{m_i,\, ts_j,\, it_k,\, cma_l,\, pr_n^m,\, f_o,\, d_p\}$, where
  - m_i denotes the mission i that is controllable and observable since the DES supervisor can change the mission (if it is required) and can observe the current mission of the team.
  - ts_j designates the team structure j that can be controlled and observed by the DES supervisor since it is one of the elements of the team configuration that the high-level supervisor makes decisions on.
  - it_k designates the information topology k that the DES supervisor observers and changes according to the mission and team structures.
  - cma_l denotes the communication architecture l that the DES supervisor chooses according to the current team structure and the information topology.
  - pr_n^m is used to specify that agent m has priority level n. This is a function of the information topology and the communication architecture as described in Section 2.1. On the other hand, the DES supervisor enables or disables different priority levels for each agent based on the health status (healthy or faulty) of that agent. This enablement or disablement is what in turn restricts the available team configurations.
  - f_o represents the occurrence of a fault o in an agent. This event is not observable by the DES supervisor (the faults are not realized until they are detected by the fault detection and isolation module). Neither can the DES supervisor control the occurrences of faults.
  - d_p represents the detection of the fault p. This is observable by the DES supervisor, however the DES supervisor cannot control its occurrence since it depends on the existing fault and the fault diagnosis system.
- q_0 denotes the initial state, and
- Q_m is the set of desired states (e.g. UAVs configurations that the team can accomplish in the mission).

To summarize, the set of observable and unobservable events are denoted by $\Sigma_o = \{m_i,\, ts_j,\, it_k,\, cma_l,\, pr_n^m,\, d_p\}$ and $\Sigma_{uo} = \{f_o\}$, respectively. The set of controllable and uncontrollable events, on the other hand, are denoted by $\Sigma_c = \{m_i,\, ts_j,\, it_k,\, cma_l,\, pr_n^m\}$ and $\Sigma_{uc} = \{f_o,\, d_p\}$, respectively.

Each state of the DES model represents a snapshot of the multi-agent system. In each snapshot, the team of agents has a particular configuration that is characterized by the specific mission, team structure, information topology, communication architecture, agents' associated priority levels, (potential) faults and fault detection information. Since the objective is to lead the team of UAVs to states where the team has a feasible configuration, these DES states are marked (they are known as the *marked states*) denoting the desired destinations in the DES machine.

The complete DES model for fault diagnosis and recovery in a team of UAVs cannot be designed in a single step due to its size. Therefore, the DES model is constructed by using multiple DESs of lower complexities. Each of these DESs represents one or a couple of logics of the low-level component behaviors as described in detail in [21]. Subsequently, these DESs are synced together to construct the complete DES model. The complete DES model symbolizes the relationships among all the low-level component behaviors and their statuses.

The fault information that is received from the low-level fault diagnosis module is used as a specification in the DES supervisory control design. This specification contains the information about the faulty UAV(s) and the desired changes in the team architecture. According to a theorem in [26], if this specification is controllable, normal, and $L_m(G)-$closed, there exists a feasible nonblocking supervisor such that the DES plant under supervision satisfies the given specification. Consequently, by taking advantage of the priority levels, a desired specification is defined such that the team of UAVs does not switch to those configurations in which the faulty UAV has a high priority level.

2.3 DES Supervisor Design

As indicated above, the DES supervisor for the hybrid fault recovery module attempts to obtain the highest achievable team performance by removing (disabling) the team configurations in which the faulty agent(s) has(have) a large impact on the team performance due to its(their) role in the team structure, the information topology, and the communication architecture. This is to be accomplished by requiring the least amount of change in the system configuration as shown in Fig. 2. In this approach based on the diagnosed fault, the DES supervisor utilizes communication topology, information topology, or team structure to change the team configuration based on the current team configuration and the desired specifications such that the influence of the fault is minimized on the team. When compared to traditional methods that are available in the literature, the proposed solution will be shown subsequently to have the capability of preventing mission failure in presence of a faulty UAV in the team. It can be shown that the necessary and sufficient conditions for existence of a fault recovery solution for the team of UAVs are the same as the required conditions for existence of a supervisor. Details are not included due to space limitations.

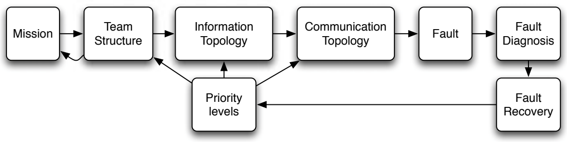

Fig. 2. Fault recovery solution for a team of multi-agents.

Note that in cases where multiple marked states (destinations) exist where each satisfies the desired specifications and where multiple paths to reach them are available, one needs to consider other criteria to select the optimal and a unique path. In the next section, by further taking into account the transition costs of the team of UAVs reconfigurations (which is defined in terms of changes in the team structure, the information topology, and the communication structure), the cooperation among the UAVs is optimized through the high-level supervisory control for arriving at an optimal solution.

2.4 DES State Costs

The high-level DES supervisory control scheme can be constructed by following through the procedure that was described in the previous section. In other words, low-level controllers are designed for achieving desired configurations that one would like the faulty UAVs to lead to (the DES marked states). By considering the cost function of each of these states as well as the path costs to reach to these states, the DES supervisor attempts to minimize the total cost of the team. At the first stage the DES supervisor does not change the low-level controller parameters to reduce the overall team cost. However, subsequently the DES supervisor may be used to fine tune the controller parameters by assuming the knowledge of the team information.

In a DES model the desired modes are represented by marked states. In marked states optimum low-level controllers are designed for each individual agent. These controllers might be modified or adjusted according to the particular configuration and/or the fault occurrence in the current state. The low-level team cost function with the desired controller is assigned as the cost of the marked state. The unmarked states do not have any cost assigned to them due to the transient nature of these states. The cost function of the low-level controllers for the team of UAVs is transferred into the high-level decision making module as the state cost. In other words, let us define the characteristic function $\chi : Q \rightarrow [0, \infty)$ which assigns a positive weight to each state $q_i, i = 1, \ldots, n$, (n denotes the dimension of the UAVs states) according to

$$\chi(q_i) = \begin{cases} J_{q_i} & \text{if } q_i \in Q_m \\ 0 & \text{if } q_j \notin Q_m \end{cases} \tag{1}$$

where J_{q_i} is the total cost function of the low-level controller. This state cost is obtained by summing the upper bounds of the cost functions of the low-level controllers for the team of UAVs with a particular configuration at state q_i. In other words, $J_{q_i} = \sum_{k=1}^{N} |J^{q_i}_{UAV_k}|$, where $J^{q_i}_{UAV_k}$ is the cost function associated with the controller that is designed for the UAV_k ($k = 1, \ldots, N$) at the team configuration corresponding to the DES state q_i.

It should be noted that in the special case when the low-level controller of UAV_k does not change at different team configurations, the controller cost does not change at different DES states either. In other words, $J^{q_1}_{UAV_k} = \ldots = J^{q_{N_q}}_{UAV_k}$, where N_q is the total number of DES states. Moreover, if none of the UAV controllers change at different configurations all the marked states will also have the same state cost, that is $\forall q_i, q_j \in Q_m, J_{q_i} = J_{q_j} = \sum_{k=1}^{N} |J^{q_i}_{UAV_k}|$.

2.5 DES Event Costs

In a DES model each event corresponds to a change in the multi-agent team configuration. These events might be due to changes in the mission, the information topology, the communication architecture, or the team structure. Consequently, these events will carry an associated cost to the team. Costs will be assigned to each event in the DES model. The event cost in a DES model can be formalized as follows [10]: The cost associated with each event is denoted by $\tilde{\pi} : \Sigma^* \rightarrow [0, \infty)$ such that $\forall \sigma_k \in \Sigma, \forall s \in \Sigma^*$, we have (a) $\tilde{\pi}(\sigma_k) = \tilde{\pi}_k \in [0, \infty)$; (b) $\tilde{\pi}(\epsilon) = 0$; and (c) $\tilde{\pi}(\sigma_k s) = \tilde{\pi}(\sigma_k)\tilde{\pi}(s)$, where ϵ is the empty string in the DES. Next, we quantitatively evaluate the resources that are required for different configurations in a team of UAVs. These low-level costs are transferred to the high-level module as the event costs. A possible method for calculating these costs for the application of a team of UAVs is provided below. It should be noted that our methodology for the optimal hybrid fault recovery does not depend on how these costs are precisely calculated. These low-level costs may be computed and evaluated according to other preferred methods and based on the specific application at hand.

We have the following event costs. $\tilde{\pi}(ts_j)$ denotes the associated cost for the team structure switching from one configuration to another that can be calculated based on the total distance that the UAVs need to traverse for the reconfiguration in the team, which can be defined according to the leader aircraft [6], $\tilde{\pi}(it_k)$ denotes the power required for computation of different information topologies that can be determined based on the computational complexity, which is a function of the number of states that need to be exchanged, and the employed CPU's power consumption (thermal design power of the CPU), $\tilde{\pi}(cma_l)$ denotes the power consumption of the communication architectures that can be calculated based on the utilized wireless equipments and the number of required communication links. For instance, a light commercial long range (≈ 1.6 Km) wireless transmitter like RCATS, which is employed in UAVs, consumes 3 (W) of power [18]. Therefore, the total communication cost can be obtained by multiplying the number of required communication links with the power each of these transmitters use, $\tilde{\pi}(pr_n^m)$ denotes that there are no costs associated with the priority levels since these events are the result of information topologies and communication architectures whose costs have already been determined, $\tilde{\pi}(f_o)$ denotes that there are no costs associated with the occurrence of a fault. Since this event is not observable by the DES supervisor therefore the supervisor does not make decisions based on the specific fault occurrence. However, it should be noted that the occurrence of a fault does change the state costs in equation (1) based on the performance of the current controller in the faulty condition, and finally, $\tilde{\pi}(d_p)$ denotes the costs associated with the detection and identification of a fault that can be evaluated based on the characteristics of the fault and its impact on each UAV and the entire team. In this paper, we consider the icing problem as the structural fault in each UAV, therefore the icing fault impacts performance of the low-level controllers. In other words, the DES events do not have any specific associated costs, however, the DES state costs ($\chi(q_i)$ that is given by equation (1)) are changed as a result of the fault.

2.6 DES Transition Costs

It should be noted that the cost of the above introduced events might differ when they occur at different DES states (situations). Thus, we need to calculate the transient costs according to the *origin* and the *destination* states, and the event costs between these two states. One is also required to represent the faults in the high-level DES model as unobservable events. These faults have different occurrence *probabilities* at each state. These probabilities are to be multiplied by the potentially higher event costs. For simplicity in our presentation, let us assume that there is only a single fault that is present in the team at any given time. It should be emphasized that our concept of transition cost below can be easily extended to multiple concurrent faults. These results are not included here due to space limitations.

The state transition cost of the DES model is defined as the function $\pi(q_j, q_k) : Q \times Q \rightarrow [0, \infty)$ corresponding to the state q_j and state q_k ($j \neq k$) such that $\pi(q_j, q_k) = \infty$ if $\{\sigma \in \Sigma : \delta(q_j, \sigma) = q_k\} = \emptyset$ and $\pi(q_j, q_k) = \sum_{i=1}^{n_f+1} \Big(P_{f_i|q_j} * \pi_{jk}^{f_i} + (1 - P_{f_i|q_j}) * \sum_{\sigma \in \Sigma^* : \delta(q_j, \sigma) = q_k} \tilde{\pi}(\sigma)\Big)$, otherwise, where $P_{f_i|q_j}$ is the probability that the fault f_i occurs at the state q_j, and $\pi_{jk}^{f_i}$ is the state transition cost at the state that the fault f_i leads to, and n_f denotes the number of faults.

For sake of notational simplicity, we rewrite $\pi(q_j, q_k)$ as π_{jk} in our subsequent analysis. The probability $P_{f_i|q_j}$ depends on the characteristics and properties of different missions, team structures, information topologies, and communication architectures. This probability may be calculated according to the worst case scenario in each state by invoking an *a priori* knowledge about the team. The state transition cost matrix, that is denoted by $\mathbf{\Pi}$, is accordingly defined for the DES model G having n_G states with zero diagonal terms and π_{ij}, $i = 1, \cdots, n_G$, $j = 1, \cdots, n_G$, $i \neq j$ for the off-diagonal terms. The state transition cost matrix can be transformed into a weighted graph by employing it as the adjacency matrix of the graph.

2.7 Optimal Recovery by Using Probabilistic DES

Next, we invoke combinatorial optimization methods, as described in [7, 9], to determine the shortest path between any two points in a weighted digraph. These methods are considered due to their capability in finding the minimum cost in a weighted digraph, as opposed to using, for example the method that is proposed in [10] for choosing the best string of events in the DES (*DES language*) when there exist more than one path. Among the combinatorial optimization methods the Dijkstra algorithm can be selected to determine the minimum cost path in the resulting weighted digraph. The optimal path in the digraph in fact represents the reconfiguration in which the team of UAVs has indeed the minimum cost in terms of changes in the team configurations (team structure, information topology, and communication architecture). In this digraph, the desired destinations (team of UAVs' configurations) are selected based on the cost of UAVs low-level controllers in those configurations which are the marked states with the lowest state costs.

It should be pointed out that in case all the UAVs are similar, the problem formulation will be simplified and the high-level DES representation will also be accordingly more simplified. Specifically, the number of the

DES states will be reduced due to the fact that all the agents have similar property and there is no difference among different team structures before any of the agents become faulty. Consequently, the generated graph from the simplified DES state machine would have fewer edges and vertices. Accordingly, the optimization algorithm would run faster to find the shortest path in the graph.

3 Case Study and Simulation Results

To illustrate and demonstrate the concepts, methodologies, and design procedures that are introduced in this work for a team of multi-agent systems, consider a team of five UAVs and let the mission be defined as that of flying en-route at a constant altitude to take images from an assigned area of interest. The team structure is assumed to be V-shaped in the z-plane as shown in Fig. 3 to reduce the agents fuel consumption. The agents may be positioned at different locations as they fly, creating different team structures. It is assumed that a structural fault (due to icing accumulation on the UAV wings and body) may occur during the flight. The icing is a well-known effect in the aircraft industry and is considered as a major cause for structural parameter changes [3], [11], [13], [14], [15], and [16]. It is therefore necessary to model the icing as a structured parametric uncertainty.

Due to the icing phenomenon, airflow around the wings becomes noticeably distorted. Several negative effects are manifested, including a loss in the lift, an increase in the drag, and a lower stall angle, that will all lead to deteriorations in the stability of the aircraft. The icing factor can be modeled by applying the aircraft parameters such as the air speed and dimension of the wing and its shape to a software that was developed in [27]. In the low-level module, two robust sliding mode controllers are designed. Specifically, one controller that has and the other controller that does not have the knowledge of the icing severity factor which is assumed to be obtained from the diagnostic module. The aircraft can actually become *unstable* due to presence of full icing if the implemented controller is designed based on the assumption of no icing (clean condition) [22].

3.1 High-level DES Modeling and Development

In Fig. 3, the information topology that is depicted corresponds to the decentralized virtual structure [1], where each agent communicates with only its two nearest neighbors such that the team forms a loop of connected agents as shown in the first team structure (labelled as TS_1). In the second team structure TS_2, on the other hand, the physical inter-UAV communication range constraints around each UAV are explicitly indicated. The team structure TS_2 also represents a possible communication architecture. Therefore, UAV_1 and UAV_5 depend on both the UAV_4 and UAV_3 for communicating their commands to each other. Since the information topology considered is a decentralized virtual structure, all the agents have the same influence level on the team as far as the information topology is concerned $\big($in other words, $IT_{UAV_1} = IT_{UAV_2} = \ldots = IT_{UAV_5}\big)$. However, the communication architecture has 2 different influence levels, namely one level for the UAVs communicating with 3 other UAVs (that is UAV_3 and UAV_4 in TS_2) and the other level for the UAVs communicating with 1 or 2 other UAVs $\big($that is $CMA_{UAV_3} = CMA_{UAV_4} > CMA_{UAV_1} = CMA_{UAV_2} = CMA_{UAV_5}$ in $TS_2\big)$. In fact, UAV_3 and UAV_4 have influence on the entire team and their failure would result in the team communication failure and accordingly the team failure. In contrast, communication failure of the other agents only result in their own individual failure and not the team failure. Finally, the two levels of priority can be defined as shown in Fig. 3 where $Pr_1 > Pr_2$.

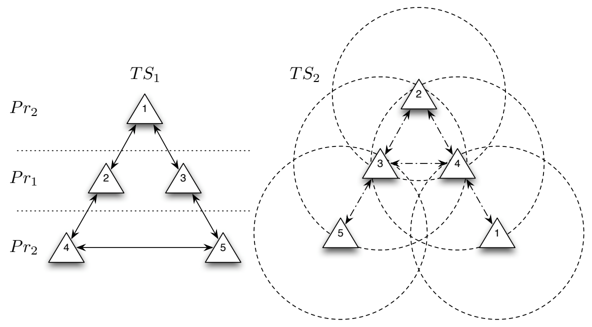

Fig. 3. A team of five UAVs performing a mission under two team structures TS_1 and TS_2.

In order to demonstrate in more detail how our optimal fault recovery solution works, let us consider the team model without the priority levels. In the resulting system, the UAVs can choose between two alternative team structures (namely TS_1 and TS_2) as shown in Fig. 3. Moreover, the UAVs information topologies can, as an example, be either the leader-follower (IT_1) or the decentralized virtual structure (IT_2) as shown in Fig. 4-(a) (for more details on these topologies refer to [1]). Accordingly, the UAVs can communicate through two different communication architectures (for example, CMA_1 and CMA_2 for the leader-follower and the decentralized virtual structure, respectively) as depicted in Fig. 4-(b). The DES model consisting of 8 states for the mission described earlier without a fault (corresponding to the nominal or normal operational condition) is shown in Fig. 5-(a).

Now, let us assume that only the flight control system of UAV_2 is equipped with the low-level fault recovery solution that was proposed in [22]. The L_2 norm tracking errors (selected as "cost" functions) have been obtained for the UAV_2 corresponding to a conventional robust controller design as well as the proposed control method in [22]. The results are obtained under the assumption of the worst case icing (full icing) [3]. Using the proposed control in [22] where the icing estimates are utilized directly in the controllers have also been obtained which show a significant reduction in the L_2 norm tracking errors. The low-level fault recovery results (not included due to space limitations) show that the L_2 norm error

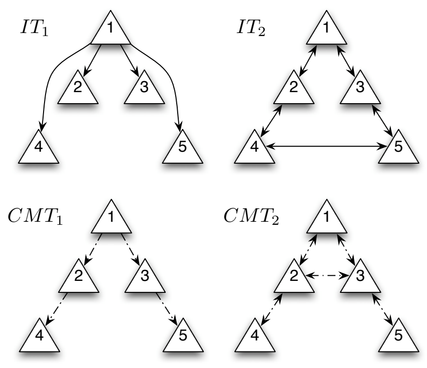

Fig. 4. (a) (Top Graph) Two information topologies (IT) for the leader-follower and the decentralized virtual structure [1], and (b) (Bottom Graph) the corresponding two alternative communication architectures (CMA).

cost of all the UAVs except UAV_2 is $1.3e9$. The L_2 error cost of UAV_2 is obtained by using the robust controller that is designed by utilizing the icing percentage estimate and is obtained to be $7.5e7$. Therefore, the characteristic functions (as given by equation (1)) are obtained as follows: $\chi(1) = \chi(2) = \chi(6) = \chi(8) = J_{q_1} = \sum_{k=1}^{5} J^{q_1}_{UAV_k} = J^{q_1}_{UAV_2} + 4 \times J^{q_1}_{UAV_1} = 7.5e7 + 4 \times 1.3e9 = 5.2e9$. Note that the other unmarked states will have zero state costs (i.e., $\chi(3) = \chi(4) = \chi(5) = \chi(7) = 0$). The state costs are depicted in Fig 5-(b) for the normal case (no icing). Consequently, in the normal condition (no icing) all the marked states are considered as the preferred configurations for the team of UAVs. Therefore, the DES supervisor leads the team of UAVs to any of these marked states. However, since there are multiple paths to reach each of these marked states (as can be observed from Fig. 5-(a)), the optimum path can be determined by considering the associated cost for each of these paths that is described next.

Fig. 6 depicts a *portion* of the high-level fault recovery model. In this figure, the DES states corresponding to the normal behavior of the team are clustered as "Normal". When the fault f_1 that corresponds to the icing in the UAV_1 occurs the team enters a transient mode where the states are clustered as "f_1 Occurred". The states are named such that they represent the relationship among the states in the normal and the transition conditions. For instance, the team configuration in state 4 of the normal mode is the same as that of the state $4'$ in the transient mode. When the fault f_1 is detected (event d_1 is generated) the team enters a faulty mode whose states are clustered as "f_1 Detected". It can be seen from Fig. 6 that the number of the marked states X_m in the normal and the transient modes are the same ($X_m = \{1, 2, 6, 8\}$), since the fault f_1 is not observable. However, in the faulty mode there are only two marked states corresponding to the states 2 and 8 of the normal mode ($X_m = \{10, 16\}$).

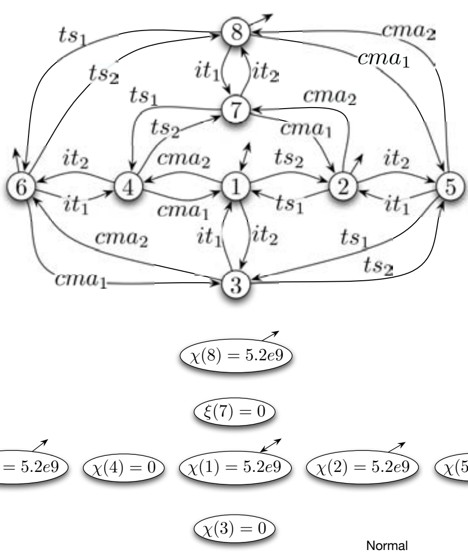

Fig. 5. (a) (Top Graph) The DES model for a team of 5 UAVs that is configured with two team structures, two information topologies, and two communication topologies under no faults, and (b) (Bottom Graph) the associated costs in each DES state.

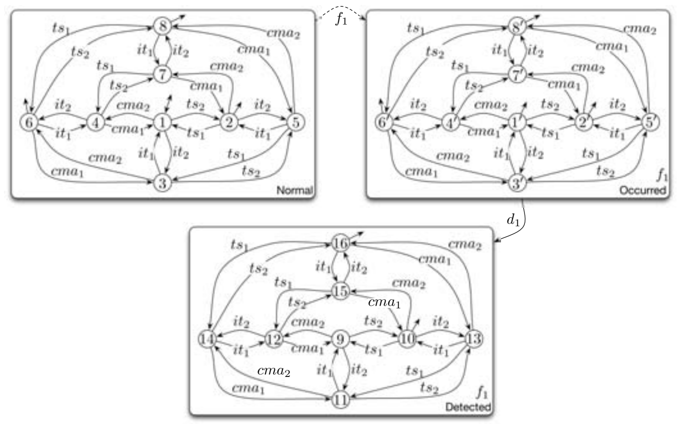

Fig. 6. A portion of the DES model for a team of 5 UAVs that is configured with two team structures, two information topologies, two communication topologies, two priority levels, and one fault (f_1) that is associated with the UAV_1 and its detection signal d_1.

3.2 Hybrid Fault Recovery Solution

Let us now assume that icing occurs only on the UAV that is on the front vertex, since according to [14, 15] the vortex effects of the front aircraft influences the ice accumulation on the following aircraft. Moreover, let us assume that none of the UAVs are equipped with any deicing facilities. Let us first construct the DES models of the team under single as well as multiple faults. When icing occurs in UAV_1 that is flying on the front vertex, the L_2 norm error cost of UAV_1 embedded with the robust controller is calculated to be $6.4e9$. Therefore, the characteristic functions (as given by equation (1)) are now modified to $\chi(9) = \chi(14) = J^{q_6}_{UAV_1} + J^{q_6}_{UAV_2} + 3 \times J^{q_6}_{UAV_3} = 6.4e9 + 7.5e7 + 3 \times 1.3e9 = 10.3e9$, where the team of UAVs has the same configuration in the DES states 9 and 14 as in the DES states 1 and 6, respectively (refer to Fig. 6), except that in the DES states 9 and 14 UAV_1 is diagnosed as the faulty UAV (f_1 is detected). The characteristic functions of the other marked states do not change in this case (i.e., $\chi(2) = \chi(8) = \chi(10) = \chi(16) = 7.5e7 + 4 \times 1.3e9 = 5.2e9$), since no icing is yet detected in UAV_2 and icing does not occur in UAV_1 when it is not flying on the front vertex. The unmarked states will always have zero state costs (i.e., $\chi(11) = \chi(12) = \chi(13) = \chi(15) = 0$).

We now consider our second scenario where we have multiple faults. A portion of the team of UAVs DES model when two UAVs are *concurrently* faulty (that is UAV_1 and UAV_2) is depicted in Fig 7-(a). It can be observed that the DES model becomes larger when compared to a single fault scenario of Fig. 6 due to the *sync* product of the DES of the normal operational condition (shown in Fig. 5-(a)) and the DES which models the detection of the faults in either of the UAV_1 or UAV_2. The modified state costs are also shown in Fig 7-(b) for the case when icing is detected in UAV_1.

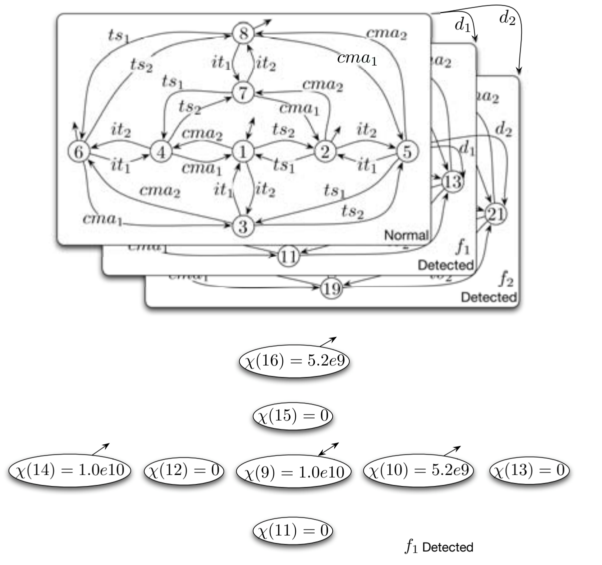

Fig. 7. (a) (Top Graph) A portion of the DES model for a team of 5 UAVs that is configured with two team structures, two information topologies, two communication topologies, two priority levels, and two faults that are associated with the UAV_1 and the UAV_2, and (b) (Bottom Graph) the associated cost of each DES state when the fault f_1 (associated with the UAV_1) is detected.

In case when UAV_2 is flying in the front vertex and the icing is detected, the L_2 norm error cost that is obtained by employing the robust controller with the icing estimate [22] is calculated to be $2.1e8$. Consequently, the characteristic functions of the DES states having similar team configurations corresponding to the DES states 2 and 8 are modified to $\chi(18) = \chi(24) = 2.1e8 + 4 \times 1.3e9 = 5.4e9$. The characteristic functions of the other marked states (having similar team configurations corresponding to the DES states 1 and 6) do not change in this case (i.e., $\chi(17) = \chi(22) = 7.5e7 + 4 \times 1.3e9 = 5.2e9$), since no icing is yet detected in the UAV_1.

Now, let us consider the complete high-level model of the team that contains all the configurations and the two possible faults in UAV_1 and UAV_2 (as it is partially shown in Fig 7-(a)). Our goals are to demonstrate the high-level fault recovery DES supervisory design as well as the advantage of our proposed hybrid optimization method. The DES model of the UAVs team is first generated by following the procedure that is described and outlined in Section 2.2 and the results that are presented in [21]. By assigning a desired specification after detection of an icing fault in the UAV_1 structure, the high-level DES supervisor can then prevent the selection of team configurations where performance of UAV_1 affects the other UAVs due to its role in the information topology and the communication architecture. In other words, the high-level DES supervisor prevents those situations where the UAV_1 has the high priority (priority level one, Pr_1). Similarly, by assigning another desired specification to f_2 that is associated with a fault in UAV_2, the high-level DES supervisor can prevent the team from switching to those configurations where the faulty UAV_2 has a high priority in the team. These two generated specifications are then synced together to construct the desired specifications for the team of UAVs in presence of a fault in either UAV_1 or UAV_2, or both.

Next, a partial observer DES supervisor is designed by invoking the partial supervisory theory in [26]. This supervisor partially observers the DES model, since faults are not observable until the detection signals (d_1 and d_2) are generated. Therefore, the supervisor's decision should not depend on the occurrence of f_1 and f_2. The supervisor prevents the team from switching to configurations where the faulty agent(s) has(have) high impact on the team performance. The resulting DES model (named G_{DES}) has 44 and 378 states and events, respectively. Due to space limitations this model is not included here.

3.3 *Optimal Hybrid Fault Recovery Solution*

The DES supervisor can provide solutions for the team configuration to be in either the marked state 18 (TS_2, IT_1, and CMA_1) or 24 (TS_2, IT_2, and CMA_2) (refer to Fig 7-(a)) when icing occurs in UAV_2. This follows from the fact that the cost function of these states ($\chi(18)$ and $\chi(24)$) are less than that of the other marked states. Similarly, the DES supervisor leads the team to the DES states 10 and 16 when icing is only detected in UAV_1. On the other hand, since the transition cost from the initial DES state 1 to the DES state 18 (with the total transition cost $= \pi_{12} + \pi_{218} = \pi_{117} + \pi_{1718} = 3.1e5$) is less than the transition cost to state 8 (with the total transition cost $= \pi_{14} + \pi_{47} + \pi_{78} + \pi_{824} \approx 3.4e5$), the supervisor leads the team to the marked state 18 whenever the icing fault occurs in UAV_2 or in both UAV_1 and UAV_2.

A corresponding state transition matrix cost (denoted by $\mathbf{\Pi}_{G_{DES}}$) for the G_{DES} can be constructed (this is not shown here due to the large size of the 44×44 matrix $\Pi_{G_{DES}}$). The optimum path from each state to another one can be determined by using the Dijkstra's algorithm with the Fibonacci heap [7] and [9]. The output of the optimization algorithm includes the minimum path from the current state to any other preferred marked state (the marked states with lower DES state costs). For instance, the chosen path from state 1 to state 24 (one of the marked states) is through the following states $1 \rightarrow 5 \rightarrow 10 \rightarrow 18 \rightarrow 24$, that corresponds to the sequence of high-level commands $cma_2 \rightarrow ts_2 \rightarrow cma_1 \rightarrow it_1$ with the total cost of $3.7e6$ (mW) (details not provided due to space limitations). Another example of a selected path where a fault is detected in UAV_1 (DES event d_1 has occurred) is $1 \rightarrow 2 \rightarrow 8 \rightarrow 14 \rightarrow 21$, that corresponds to the command sequence $ts_1 \rightarrow d_1 \rightarrow it_2 \rightarrow cma_2$. It can be shown that after the fault in UAV_1 is detected (the DES event d_1 has occurred) the information topology and the communication architecture are changed from it_1 to it_2 and from cma_1 to cma_2, respectively, in order to reduce the impact of UAV_1 on the other team members. Fig. 8-(a) depicts the UAV_2 states errors when the icing occurs in the leader (UAV_1) in TS_1 and the DES supervisor is not allowed to change the team configuration, although it can change the low-level controllers of UAV_2. It can be seen from this figure that the high-level supervisor tries to reduce the error by adjusting the low-level controllers in UAV_2.

In contrast, Fig. 8-(b) depicts the state errors of UAV_2 when the icing occurs in the leader in TS_1 (UAV_1) and then the DES supervisor changes the team configuration by selecting an optimal path (according to the Dijkstra solution) and assigns UAV_2 as the leader and positions it in front of the vertex.

Note that if the Dijkstra's algorithm is not utilized one may select a non-optimal path in the DES model. In this case, one needs to use another DES machine to limit the maximum number of high-level DES supervisor commands that are invoked, as there is no explicit limit on the total number of high-level commands. For instance, if no upper bound is imposed on the number of commands, a path from the same source and destination states (that is from the state 1 to the state 24) and without any cycle can yield a total cost of $1.6e09$ (mW) for the randomly selected path $1 \rightarrow 2 \rightarrow 9 \rightarrow 6 \rightarrow 14 \rightarrow 8 \rightarrow 15 \rightarrow 10 \rightarrow 18 \rightarrow 25 \rightarrow 33 \rightarrow 24$. Simulations do indeed show (not included due to space limitations) the effects of such a non-optimal path on UAV_2 when UAV_1 was the leader and is faulty. It can be concluded that the hybrid recovery solution takes longer to switch to the correct configuration (at the DES state 24 as compared to the results shown in Fig. 8-(b)).

To summarize, we have shown that our proposed optimal hybrid fault recovery methodology does not only reduce the impact of the faulty UAV on the other UAVs by reconfiguring the team architecture, but it also recovers the team from faulty conditions faster than conventional methods by going through the optimal path which is found through the use of the Dijkstra's algorithm.

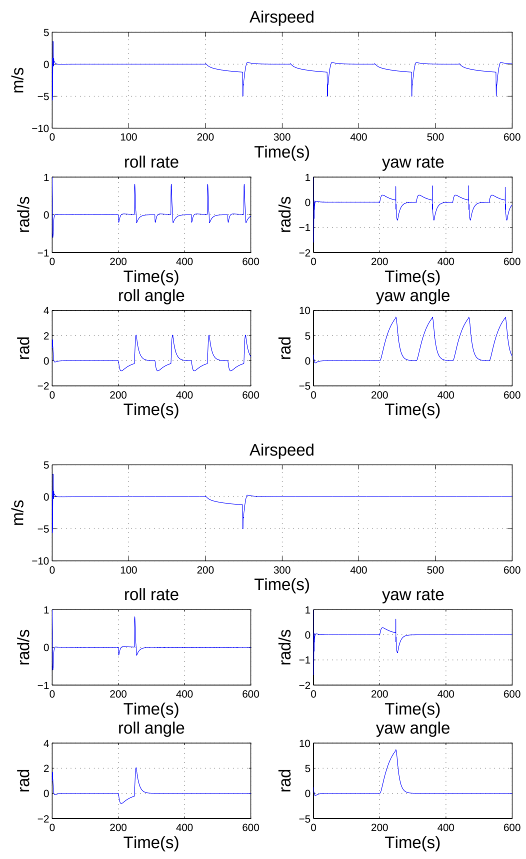

Fig. 8. (a) (Top 3 Graphs) Output errors of UAV_2 without the hybrid reconfiguration when the full icing occurs at 200 (s) in the leader although the high-level tries to reduce the error by adjusting the low-level controllers, and (b) (Bottom 3 Graphs) output errors of the UAV_2 with the hybrid fault recovery when the full icing occurs at 200 (s) where the high-level module diagnoses the fault and changes the team configuration at about 250 (s).

3.4 *Computational Comparison of the Optimal Hybrid Solutions*

In the high-level fault recovery module it can be shown that the complexity of our proposed method for determining the minimum path (Dijkstra algorithm with Fibonacci heap) is $O(m_G + n_G \log n_G)$, which is a function of both the number of states (n_G) and events (m_G). However, in the approach that is proposed in [10], the complexity is $O(n_G^4)$, which is only a function of the number of states. In most cases, the number of DES events is larger than the number of DES states; however, m_G is usually much less than $n_G(n_G^3 - \log n_G)$, implying that the complexity of our proposed approach is generally much less than the complexity of the method proposed in [10]. Therefore, our proposed method allows the reconfiguration of the team to be generated online due to its expected "low" required computational resources.

4 Conclusion

In this work, an optimal hybrid fault recovery methodology is proposed and developed for a team of unmanned aerial vehicles (UAVs). This was achieved by utilizing the cooperative nature of the team in order to accomplish desired mission requirements in presence of faults/failures in the vehicles. Using the DES framework and the supervisory control design methodologies, a high-level fault recovery module is developed and implemented for the team when two of the vehicles are faulty due to the icing (structural fault). Moreover, it is shown that the DES supervisor manages and leads the team among various configurations towards an optimal solution to reduce the effects of the faulty UAVs on the overall team mission. This is accomplished while the team reconfiguration costs and the team performance are optimally achieved. Finally, the computational complexity of the proposed optimal high-level fault recovery solution is shown to be significantly lower than another DES optimization technique that is available in the literature.

References

[1] R. W. Beard, J. Lawton, and F. Y. Hadaegh. A coordination architecture for spacecraft formation control. *IEEE Transactions on Control Systems Technology*, 9(6):777–790, November 2001.

[2] J. D. Boskovic, S. E. Bergstrom, and R. K. Mehra. Retrofit reconfigurable flight control in the presence of control effector damage. In *Proc. of American Control Conference*, pages 2652–2657, June 2005.

[3] M. B. Bragg, T. Hutchinson, J. Merret, R. Oltman, and D. Pokhariyal. Effect of ice accretion on aircraft flight dynamics. *38th AIAA Aerospace Sciences Meeting and Exhibit*, (2000-0360), Jan 2000.

[4] D. W. Casbeer, D. B. Kingston, R. W. Beard, and T. W. McLain. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science*, 37(6):351–360, May 2006.

[5] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Springer, 2nd edition, 2007.

[6] R. Fierro, C. Belta, J. P. Desai, and V. Kumar. On controlling aircraft formations. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 1065–1070, 2001.

[7] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag. Adaptive fastest path computation on a road network: a traffic mining approach. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, Vienna, Austria, 2007.

[8] D. Ionescu. *11th International Conference on Analysis and Optimization of Systems Discrete Event Systems*, volume 199, chapter Conditions for optimization of discrete event systems using temporal logic models. Springer Berlin / Heidelberg, 1994.

[9] B. Korte and J. Vygen. *Combinatorial Optimization, Theory and Algorithms*, volume 21 of *Algorithms and Combinatorics.* Springer, 4th edition, 2008.

[10] l. Chattopadhyay and A. Ray. Language-measure-theoretic optimal control of probabilistic finite-state systems. *International Journal of Control*, 80(8):1271–1290, August 2007.

[11] A. Lampton and J. Valasek. Prediction of icing effects on the lateral/directional stability and control of light airplanes. In *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, number 2006-6834, Keystone, Colorado, Aug 2006.

[12] R. Mohammadi, S. Hashtrudi-Zad, and K. Khorasani. A hybrid architecture for diagnosis in hybrid systems with applications to spacecraft propulsion system. *Proc. 2007 IEEE International Conference on Systems, Man and Cybernetics*, pages 3184–3190, October 2007.

[13] Technology Office of the Under Secretary of Defence for Acquisition and Logistics. OSD UAV reliability study. www.acq.osd.mil/uas/, Feb 2003.

[14] E. O. Ogretim. *Investigation Of Relative Humidity And Induced-Vortex Effects On Aircraft Icing.* PhD thesis, West Virginia University, 2006.

[15] E. O. Ogretim, W. W. Huebsch, J. Narramore, and B. Mullins. Investigation of relative humidity and induced-vortex effects on aircraft icing. *AIAA Journal of Aircraft*, 44(6):1805–1814, 2007.

[16] W. Olsen, R. Shaw, and J. Newton. Ice shapes and the resulting drag increase for a naca 0012 airfoil. *NASA technical memorandum 83556*, 1984.

[17] M. A. Peot, T. W. Altshuler, A. Breiholz, R. A. Bueker, K. W. Fertig, A. T. Hawkins, and S. Reddy. Planning sensing actions for UAVs in urban domains. In Edward M. Carapezza, editor, *Unmanned/Unattended Sensors and Sensor Networks II*, volume 5986, pages 59860J.1–59860J.10. SPIE, 2005.

[18] RCAT Systems. Rcats 11 channel airborne telemetry systems. www.rcatsystems.com, 2006.

[19] A. Saboori and S. Hashtrudi-Zad. Fault recovery in discrete-event systems. In *Proceedings of Computational Intelligence: Methods and Applications, CIMA'05 (ICSC/IEEE)*, Istanbul, Turkey, Dec 2005.

[20] R. Sengupta and S. Lafortune. An optimal control theory for discrete event systems. *SIAM journal on control and optimization*, 1998.

[21] M. M. Tousi, A. G. Aghdam, and K. Khorasani. A hybrid fault diagnosis and recovery for a team of unmanned vehicles. *Proc. 3rd IEEE SMC International Conference on System of Systems Engineering*, pages 1–6, June 2008.

[22] M. M. Tousi and K. Khorasani. Fault diagnosis and recovery from structural failures (icing) in unmanned aerial vehicles. In *Proc. of IEEE Systems Conference*, pages 302–307, March 2009.

[23] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A review of process fault detection and diagnosis part I: Quantitative model-based methods. *Computers and Chemical Engineering*, 27:293–311, March 2003.

[24] S. R. Verret and S. Monckton. Multi-unmanned vehicle systems (nUxV) at Defence R&D Canada. In G. R. Gerhart, C. M. Shoemaker, and D. W. Gage, editors, *Unmanned Systems Technology VIII*, volume 6230, pages 62301P1–10. SPIE, 2006.

[25] X. Wang and A. Ray. A language measure for performance evaluation of discrete-event supervisory control systems. *Applied Mathematical Modelling*, 28(9):817–833, 2004.

[26] W. M. Wonham. *Supervisory Control of Discrete-Event Systems.* Systems Control Group, Edward S. Rogers Sr. Dept. of Electrical and Computer Engineering, University of Toronto, Canada; available at http://www.control.utoronto.ca/DES, 2009.

[27] W. B. Wright. A summary of validation results for LEWICE 2.0. *Technical report, NASA*, Dec 1998.