Review of Tools and Methods for Fault Tolerance in Spacecraft Mission Planning

Ali Nasir, Ella Atkins, Ilya Kolmanovsky

***Abstract*—Due to the hostile nature of the space environment, fault tolerance has become a key part of recent research in spacecraft mission planning. Spacecraft are complex systems; mission planning in spacecraft may not just involve text book Artificial Intelligence based approaches. Requiring additional fault tolerance capability makes the problem more challenging and diverse. This paper reviews the relevant tools and methods from Artificial Intelligence and Control Systems theory for fault detection and diagnosis. Different types of modeling approaches have been discussed followed by control and mission planning. Existing approaches are evaluated and gaps have been identified. Possible approaches to fill those gaps are discussed.**

***Index Terms*—Spacecraft mission planning, fault tolerance, timeline based planning, plan database.**

I. Introduction

MISSION planning is a challenging task in general. Planning is perhaps most challenging for spacecraft because the spacecraft cannot be modified or repaired once launched. Further, thermal cycles, radiation, and scattered debris in the space environment represent substantial hazards to spacecraft. A spacecraft may not be in continuous contact with a ground station. This means that if something goes wrong during the mission, the spacecraft is on its own until communication with the ground station becomes available. This motivates the incorporation of fault tolerance into spacecraft mission planning.

There are two major approaches for achieving fault-tolerance in spacecraft missions from the literature. One method is to calculate a preset response for every possible situation while the second is to perform online planning based on predefined guidelines [1]. A collection of preset responses for the spacecraft mission defines a "plan database" where each "plan" is a sequence of actions that leads to the completion of a specific goal-achieving or recovery task. Generation of a plan onboard the spacecraft offers flexibility but can require nontrivial computational resources. Many techniques have been proposed to reduce planning complexity. For example, iterative repair [2] methods focus on improving plan quality as time permits. Distinction of short and long-term planning facilitated scheduling in Hubble Space Telescope [3]. Besides planning for normal situations, planning to manage faults has also been studied [4][5]. In case of a fault, the current plan of the spacecraft typically becomes invalid due to change in system performance and capabilities. A new plan can be formulated via online deliberation during which the spacecraft is typically placed in "safe mode" hence causing delay in mission completion as a minimum and missed opportunities in some cases, e.g., a fly-by that cannot be repeated. A plan database can also contain pre-calculated plans to manage each fault anticipated by system engineers in advance. However, since spacecraft faults cannot all be anticipated in advance, there is a limit to having plans ready in the database to be activated when required.

To-date, no plug-and-play software has been developed for spacecraft mission planning. However, some tools have been developed to assist in developing mission plans for multiple missions. For example, ASPEN [6] and CASPER [2] include structures that allow for inclusion of activities, states, constraints, and contingency responses. SPIKE [3] is another example of an intelligent scheduling framework. Spacecraft mission planning typically requires information from astrodynamics, science payload management, spacecraft kinematics and dynamics, fault diagnosis, fault tolerance, and constraint satisfaction. Therefore it is important to understand how all these systems work to develop a consolidated mission planning framework. For example, consider a spacecraft mission of collecting scientific data from astronomical objects. Each object may not be accessible to the spacecraft payload equipment from every point in the orbit. Therefore mission planning needs to have information about the orbital dynamics. Secondly, once an astronomical object of interest is "visible" to the payload, mission planning has to incorporate constraint such as sun avoidance, available power, and health of the spacecraft components. Specifically, if a stable and accurate attitude pointing is not guaranteed (due to faults in any of the attitude sensors or actuators) it would be difficult to collect scientific data. Finally, if multiple astronomical objects are visible at the same time, spacecraft planner must decide the sequence of data collection.

This paper provides a review of technical literature relevant

Submitted on 7th February, 2017, revised on 30th July, 2018.

Ali Nasir is with Electrical Engineering Department, Faculty of Engineering, at the University of Central Punjab, Lahore, Punjab 54782 Pakistan (e-mail: a.nasir@ucp.edu.pk).

Ella M. Atkins is with the Department of the Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA (e-mail: ematkins@umich.edu).

Ilya V. Kolmanovsky is with the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA (e-mail: ilya@umich.edu).

to fault tolerance in spacecraft mission planning. Section II presents an overview of the mathematical modeling techniques used in the spacecraft domain. Section III presents a review of spacecraft mission planning and motion estimation and control techniques. Section IV presents references and methods in hybrid systems or switched control systems that have been successfully used for fault tolerant control. This section also presents methods in Markov Decision Processes (MDPs) to support planning under uncertainty. Section V describes existing frameworks for fault tolerant attitude planning in artificial intelligence and in control systems theory. Representation gaps and possible solutions have also been discussed in the same section. Concluding remarks and future directions are discussed in Section VI.

II. Spacecraft Modeling

Before moving into the discussion of spacecraft mission planning, it is important to understand the basic components of spacecraft. A spacecraft is a complex system with discrete as well as continuous time behavior. It consists of subsystems such as attitude and orbital control, telemetry tracking and command, power generation storage and distribution, thermal control, and structures subsystem [7]. Interconnections of various subsystems onboard the spacecraft is depicted in Fig. **1**. Various aspects of the spacecraft are modeled using different mathematical tools. For example dynamics and kinematics are modeled using the differential equations whereas discrete component switching and faults can be modeled as discrete variables. Like most of the models in real world, spacecraft models are not exact representation of actual spacecraft behavior. Inaccuracy in modeling (as well as unpredictable nature of space environment) causes uncertainty and one of the ways to model uncertainty is through Bayesian Networks. In the rest of the section we discuss two aspects of spacecraft modeling. One is the modeling of the motion of spacecraft and second is the modeling of processes and events with spacecraft.

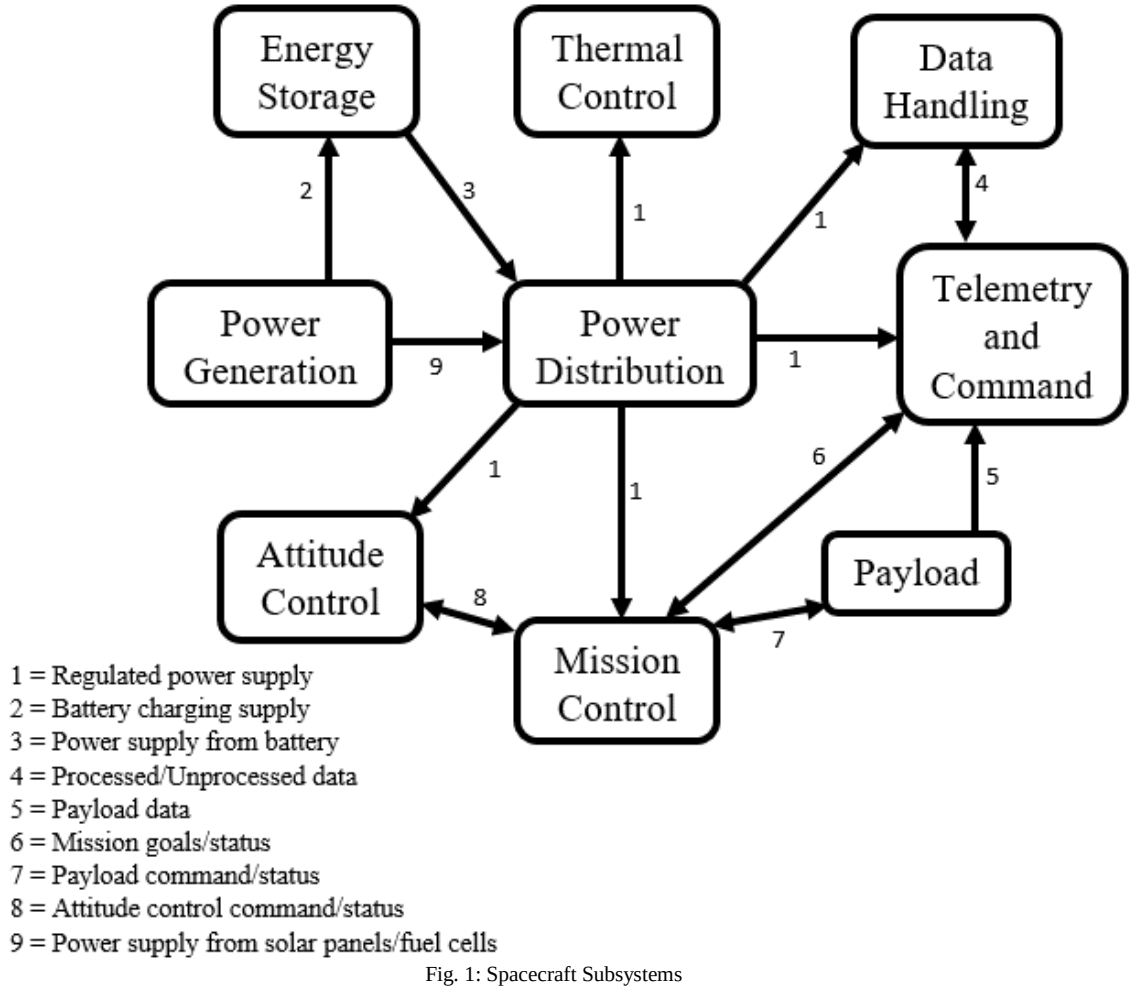

Fig. 1: Spacecraft Subsystems

A. *Modeling of the Motion of Spacecraft*

Motion of the spacecraft is of two types. One is orbital motion and second is attitude related motion. For most spacecraft mission, orbital motion is uncontrolled after the desired orbit is achieved after initial launch. Attitude however is actively controlled in majority of spacecraft. Attitude motion is represented by the equations of kinematics and dynamics of spacecraft. There are multiple ways of representing spacecraft attitude kinematics [8]. Equations (1a), (1b), and (1c) represent alternate ways to model the kinematics of the spacecraft as a rigid body.

$$\dot{q} = \frac{1}{2}(q_4\Omega - [\Omega \times q])$$

$$\dot{q_4} = -\frac{1}{2}(\Omega^T q) \tag{1a}$$

$$\dot{R} = \begin{bmatrix} 0 & \omega_3 & -\omega_2 \\ -\omega_3 & 0 & \omega_1 \\ \omega_2 & -\omega_1 & 0 \end{bmatrix} R \tag{1b}$$

$$\begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \dot{\theta_3} \end{bmatrix} = \begin{bmatrix} cos\theta_2 & sin\theta_1 sin\theta_2 & cos\theta_1 sin\theta_2 \\ 0 & cos\theta_1 cos\theta_2 & -cos\theta_2 sin\theta_1 \\ 0 & sin\theta_1 & cos\theta_1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{1c}$$

In (1a), q is a 3 × 1 vector of the first three elements of attitude quaternion of the spacecraft with respect to an inertial frame of reference; q_4 is a scalar representing the fourth element of the quaternion. Ω is a 3 × 1 vector representing spacecraft angular velocities in a body-fixed frame. In (1b), R is the 3 × 3 rotation matrix for the spacecraft whereas ω_1, ω_2, and ω_3 represent components of Ω. In (1c), $\begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^T$ represent angular velocities in the body-fixed frame and $\begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix}^T$ represent Euler angles roll, pitch, and yaw, respectively, with respect to the inertial frame. There are other ways to represent the attitude, e.g. Euler-Rodriguez parameters [8] [9] not discussed here. The quaternion representation is common for spacecraft as it has no singularities and requires only four continuous-valued quantities in its representation. Euler angles do have singularities, while rotation matrices have no singularities but must be represented with nine values.

The attitude dynamics of a rigid spacecraft are represented as

$$\begin{aligned} \dot{\omega_1} &= \frac{1}{J_1}[(J_2 - J_3)\omega_2\omega_3 + u_1], \\ \dot{\omega_2} &= \frac{1}{J_2}[(J_3 - J_1)\omega_1\omega_3 + u_2], \\ \dot{\omega_3} &= \frac{1}{J_3}[(J_1 - J_2)\omega_2\omega_1 + u_3] \end{aligned} \tag{1d}$$

In (1d), (J_1, J_2, J_3) are diagonal components of 3 × 3 inertia matrix in a body-fixed frame and (u_1, u_2, u_3) represent control inputs. Note that the inertia matrix is typically assumed to be diagonal. There are a number of ways to control spacecraft attitude [10] even with two control inputs instead of three [11]. In [11] a stabilizing feedback control strategy has been developed that is discontinuous and can achieve any possible attitude value. The results presented in [11] show that although nonlinear control techniques do not apply, a stabilizing control law can still be constructed that is based on a sequence of maneuvers. Also, for attitude determination, a number of ways have been developed to estimate the attitude from the sensor readings which may or may not provide accurate measurements, e.g. [12] where Kalman filtering has been used to estimate the attitude with gyroscopes that have both drift and bias errors.

B. Modeling of Processes and Events within the Spacecraft

Modeling of the processes and events within the spacecraft can be done with discrete time variables and logical expressions (or rules). Each process involves variables that have dependencies. One way to represent dependencies among the variables is Bayesian Network. More precisely, Bayes network [13] is a way of representing dependence relations between random variables and is used for efficient computation of joint and conditional probabilities. Bayes nets can be used for modeling the internal composition of a spacecraft by noting that failure of any component is a random event and failures of components within the spacecraft depend upon each other in a way that can be determined from the interconnection and interaction of components with one another. A Bayes net succinctly represents a conditional probability table (CPT). Bayes nets can intuitively represent CPTs associated with spacecraft fault diagnosis. For example, consider a one degree of freedom (1DOF) reaction wheel system where a battery supplies power to two electronics boards (one of which is redundant for fault tolerance) that can drive the reaction wheel. A simple Bayes net model for failure probabilities of this system is shown in Fig. 2.

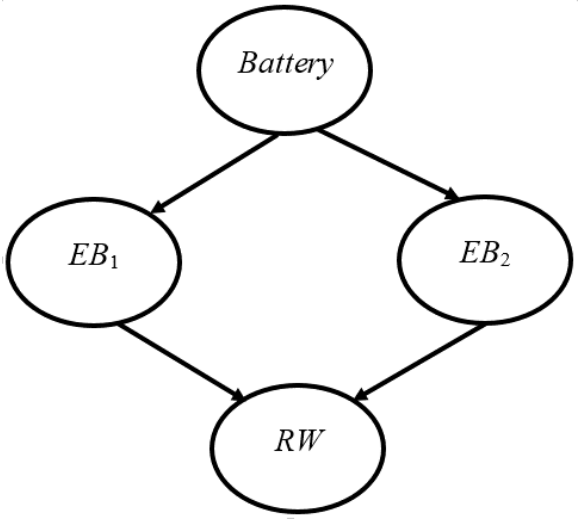

Fig. 2: Bayes net example

The above graph indicates failure of the battery affects the failure of the electronics boards (EB1 and EB2), and the failure of the electronics boards affects the functionality of the reaction wheel (RW). Furthermore, failure of the reaction wheel is conditionally independent of the failure of battery given definitive knowledge (evidence) of whether both electronics boards have failed as this evidence "breaks the dependency link" from the Batt graph node. This type of model can be used to solve for the probabilities of failure of any components or subsystems given failure information about any other component(s) or when no information or evidence is given. There are several methods for deriving probabilities from Bayes nets [13] including enumeration, variable elimination, and local propagation. The computational and memory requirements for some of the methods are shown in Table 1. In this table, *n* is the number of nodes in the Bayes Net, and all nodes are assumed to be binary, e.g. fail/not-fail. Also note that the local

propagation method has the lowest computational complexity but it is only applicable on Bayes nets that have a poly tree structure (i.e. no cycles or multiple paths connecting one node to another).

TABLE 1: COMPUTATIONAL COMPLEXITY OF METHODS FOR SOLVING BAYES NETS [13]

Method	Applicability	Memory Requirement	Computational Cost
Enumeration	general	O(n)	O($n2^n$)
Variable Elimination	general	O(2^n)	O(2^n)
Local Propagation	polytrees	O(n)	O(n)
Clustering	general	O(2^n)	O(2^n)
Conditioning	general	O(n)	O(2^n)

III. SPACECRAFT CONTROL

Once the modeling is completed, next step is to design and implement control. A spacecraft is controlled at multiple levels. At a high level, the control is in the form of mission planning and scheduling tasks. At a low level, the control refers to specification of the physical motion of spacecraft (e.g., attitude or pointing sequence).

A. Spacecraft Mission Planning and Scheduling

There are five main ingredients of the classical artificial intelligence (AI) planning problem: a finite set of discrete states, a set of state-dependent actions, the specification of desirable or goal states, the specification of an initial state, and a search method to determine an optimal sequence of actions (i.e. the solution) that leads from initial state(s) to goal state(s). For a given size of the state and action spaces, the computational complexity of finding the solution depends upon the search method used. All the planning methods are centered on the method of search that they use to find a solution. Typical real-time schedulers see the world as a set of resources and a set of resource-consuming tasks requiring up to a known worst-case utilization of each computational resource (e.g., processor or communication). Schedulers allocate resources to tasks, assigning each a start time and resource set that guarantee all deadlines are satisfied, making tradeoffs as needed to degrade best-effort tasks given resource constraints [14][15]. One of the basic algorithms used for scheduling is earliest-deadline-first scheduling [16][17] where tasks are placed in a priority queue. This means that at the occurrence of a scheduling event, (a task is finished, a new task is released, etc.) the queue is searched for the task closest to its deadline. Another basic algorithm for scheduling is rate-monotonic (RM) scheduling [18][19] where priority is given to the tasks with shortest period.

Autonomous spacecraft task planning and scheduling have been achieved for a limited set of science missions [2][20][21]. Algorithms such as iterative repair [2] have been selected due to their ability to adapt existing plans without prohibitive computational overhead. Hence, iterative repair supports modification (possibly due to occurrence of faults or change in mission priorities) and updating of a current working plan. Iterative repair adapts an existing plan by using search-based algorithms such as backtracking [22]. This results in plan improvement but optimality in general is not guaranteed by iterative repair due to the local nature of search in iterative algorithms. Reference [2] discusses continuous planning on board the spacecraft using iterative repair to support autonomous control of spacecraft. This allows for responding to the anomalies resulting in delays or resource depletion. Another reference on integrated planning and scheduling is [23] that present a Heuristic Scheduling Testbed System (HSTS). HSTS is a framework of representation and problem solving which provides an integrated planning and programming vision. HSTS involves the decomposition of continuous time domain state variables. In this manner, description and manipulation of complex resources is managed. Resources in HSTS are modeled in classical programming tasks. Schedules developed in HSTS implicitly identify a set of behaviors legal system. This is an important difference from the classical approaches; however, specify all aspects of the same behavior, nominal system. In [24], a multi-agent planning system (MAPS) for autonomous planning is built that is used to generate feasible action sequence under complex constraints. The planning model is capable of describing simultaneous activities with continuous time. The model can also handle resource and temporal constraints. The architecture of MAPS includes multiple planning agents (PAs) and a planning manager agent (PMA). One might consider the subsystems in a spacecraft as intelligent agents and the goals of the spacecraft are achieved by combined operation of the agents. Mission planning system is managed by PMA. It also functions as a communication medium between PAs. Pas interact with each other only after they have registered with PMA.

B. Spacecraft Motion Control and Estimation

Attitude control is fundamental for ensuring spacecraft stability thus requires fault tolerance. Multiple attitude control schemes are often required for implementing full reconfigurable control. A careful presentation of spacecraft attitude control methods is given in [10]. This book treats the basics of dynamic systems modeling and control. The problem of attitude stabilization and control for the rigid spacecraft under the influence of various actuators such as reaction jets, momentum wheels, and control moment gyros is discussed in this book. These techniques can provide a good support for our proposed framework especially when different types of redundant actuators are used in the spacecraft for fault tolerance. A number of stabilization and control problems for three axes stabilized as well as spin stabilized spacecraft are also treated in [10]. Specifically large-angle attitude maneuvers are discussed where spacecraft moves about an inertially fixed axis. The motion has to be as fast as possible but actuator saturation must be handled. Such maneuvers are fundamental for science data collection missions where targets of interest can only be observed by slewing the spacecraft through a sequence of large-magnitude motions. Advanced problems of spacecraft attitude control via developing logic based solutions using CMG reaction jets are treated along with momentum management for large spacecraft.

There has also been some interesting work done on

magnetic control of spacecraft attitude in [25][26][27][28][29]. For example, in [25], a magnetic dipole moment modulation based reconfigurable attitude control is presented for an earth pointing satellite. Spacecraft control with two torques has also been extensively studied e.g. in [11]. Spacecraft control with two-torques is very useful for implementing fault tolerance.

Robust spacecraft attitude estimation is required for most missions and requires fault tolerance. Most dynamics-based fault detectors are based on output estimation. One of the most useful resources in the literature of spacecraft attitude estimation is a survey paper by Lefferts et al [12]. This work presents a summary of experience in the Kalman filtering of spacecraft attitude and offers two possible implementations of the Kalman filter for systems with attitude sensors and gyros with noise terms describable by a first-order Markov process. The difference in the two schemes is only in the choice of frame for the update, for example using the complete four-component quaternion versus using the truncated quaternion where one component has been eliminated. Crassidis and Markley [30] present an attitude estimation approach based on minimization of model error (MME). The approach is designed for three-axis stabilized spacecraft. Based on the implementation example included in [30], their algorithm is shown to be robust and accurate, able to estimate attitude with or without gyro measurements. Functional form of the optimal estimation involves a linearization technique and gradient search with a linear Riccati transformation. This algorithm is demonstrated to be accurate and computationally efficient for generating state estimates based on an implementation example. Results obtained from MME-based approach indicate accurate estimation of the attitude of spacecraft using only magnetometer sensor measurements. Crassidis and Markley have also published their work on spacecraft attitude estimation using unscented Kalman filter [31] and attitude estimation using modified Rodriguez parameters [32]. Both these authors along with Cheng have published a survey on modern attitude estimation methods [33]. This survey presents a quaternion estimation filter (QUEST), extended QUEST and reverse-smoothing extended Kalman filter. In QUEST, a discrete set of sigma points is propagated and updated instead of using linearized equations for the mean and covariance.

IV. Theories and Tools Involved in Spacecraft Mission Planning

While the previous section discussed various issues involved in mission planning, in this section, the relevant theories and tools for implementing the same are reviewed.

A. Constraint Satisfaction Problems

Constraint Satisfaction Problems (CSPs) represent a class of AI planning problems where states belong to specific domains of values and there are constraints over allowable combinations of values for subsets of state variables. CSPs can be solved with algorithms that take advantage of the specific state-space formulation. A constraint satisfaction problem (CSP) involves selection of a value from a finite domain that is to be assigned to each variable in the problem, such that all the restrictions related to the variables are satisfied. Then a sequence of actions is selected that allow the achievement of goals by the plan and which satisfaction of numerical and symbolic constraints. Many combinatorial problems in operational research, such as timetabling and scheduling can be formulated as CSPs. In [34], the authors calculate the number of tree search operations involved in the solution of binary constraint satisfaction problems. It is shown experimentally and analytically that there are two principles that improve backtracking search performance i.e. placing most likely to fail first and remembering the past actions to avoid repetitions. In [35], Dechter identifies easily solvable problem classes, and has designed algorithms to calculate optimal solutions for such problems. Other useful references on CSPs by the same author include [36][37]. Brailsford et al. describe CSPs and solution techniques in [38], and also show how constraint satisfaction approach is used in solving various combinatorial optimization problems. Also the constraint satisfaction approaches of have been compared with operational research (OR) techniques e.g. simulated annealing, branch and bound, and integer programming.

B. Hybrid Systems

Hybrid systems of interest for this work contain two different types of components: continuous and discrete dynamic subsystems with interacting dynamics. Hybrid systems theory has been applied in many areas e.g. communication networks, manufacturing, engine control automotive, designing autopilot, chemical processes, and computer synchronization etc. A key role is played by Hybrid systems in the embedded control systems. Hybrid systems are also used in developing complex systems with hierarchical organization of complex systems as well as in the interaction of scheduling algorithms along with continuous and discrete intelligent autonomous systems and control algorithms. A survey on control and modeling of hybrid systems is presented in [39]. Certain characteristics of hybrid systems are highlighted in this survey. Modeling approached within hybrid systems has been illustrated through a simple three fluid-filled tank system. Further exploration of the characteristics of hybrid model is done through variations of the same example. A discussion on the analysis and control techniques for hybrid systems is also presented in [39]. Model predictive control (MPC) has also been employed on discrete time hybrid systems [40]. The cited method builds a “tree of evolution” to abstract the behavior of the hybrid system. Reachable states of the system are represented as nodes of the tree and if a transition between two states exists, the concept of a branch is utilized. Each node is associated with a cost-function value which serves as the base value for exploration of the tree. Other references on hybrid systems include [1][41]. Introduction to the theory of hybrid systems as well as discussion on its applications is included in [41]. Authors in [42] discuss the output feedback control of a class of stochastic hybrid systems.

C. Markov Decision Process (MDP)

A Markov Decision Process is a controlled Markov chain

[43] that is solved using a discrete stochastic dynamic programming (SDP) algorithm, e.g. policy iteration or value iteration [44][13]. In value iteration an expected discounted reward function is maximized that is of the form

$$V^{Pol}(s) = E\left[\sum_{t=0}^{\infty}\gamma^t R(s^t,\mu^t) \mid Pol, s^0 = s\right]. \quad (2)$$

Here, s^t represents state after t actions, and μ^t is the action applied in state s^t according to a policy Pol (s^t is a random variable). V is the value function that can also be viewed as expected discounted reward of the state. The discount factor γ ($\gamma \in (0, 1)$), indicates that the future rewards may have lower value. R is the reward function that is assumed to be finite valued. The optimal policy can be calculated as

$$Pol^*(s_i) \in \arg\max_k\left(R(\mu_k, s_i) + \gamma\sum_{j\in S} T(s_j \mid \mu_k, s_i)V(s_j)\right). \quad (3)$$

All states that are connected via any action with nonzero probability have direct relationship between their corresponding values. This relationship can be expressed using the Bellman equation [13]:

$$V_{t+1}(s_i) = R(s_i) + \max_k\left(\sum_{j\in S}\gamma T(s_j \mid \mu_k, s_i)V_t(s_j)\right). \quad (4)$$

where $V_{t+1}(s_i)$ represents value of state s_i at iteration $t+1$. $R(s_i)$ is the immediate reward of state s_i. $T(s_j \mid \mu_k, s_i)$ is the probability of transitioning from state s_i to s_j by executing action μ_k.

In terms of the convergence of value iteration algorithm, number of iterations (Itr) to reach an error bound of ε can be bounded as:

$$Itr = \left\lceil \log\left(\frac{2R_{\max}}{(1-\gamma)\varepsilon}\right) / \log\left(\frac{1}{\gamma}\right)\right\rceil. \quad (5)$$

Here ε is the required tolerance of the solution satisfying

$$\|V_{t+1}(\varsigma_i) - V(\varsigma_i)\| < \varepsilon, \forall i. \quad (6)$$

The inequality (2.3.3-5) is ensured by [13]

$$\|V_{t+1}(\varsigma_i) - V_t(\varsigma_i)\| < \varepsilon\left(\frac{1-\gamma}{\gamma}\right). \quad (7)$$

Value iteration has computational complexity of the order $O(N^2k)$ where k is the number of actions and N is the number of states in the MDP. As described in [43], Equation (4) converges to a unique solution. The solution of Equation (4) achieves its maximum value of the right hand side in Equation (2). If the policy is calculated using (3) with solution of (4), it will be optimal with respect to (2).

The value iteration algorithm used to solve (4) and find an optimal policy from (3) is shown in Algorithm 1.

Algorithm 1: Value Iteration algorithm [44]

Step 0. Initialization:
Set $V^0(s) = 0$ for all $s \in S$
Fix a tolerance parameter $\varepsilon > 0$
Set $t = 1$.

Step 1. For each $s \in S$ compute:

$$V_t(s) = R(s) + \max_{\mu\in M}\left\{-C(\mu) + \gamma\sum_{s'\in S} T(s,\mu,s')V_{t-1}(s')\right\}$$

Step 2. If:

$$\|V_t - V_{t-1}\| < \varepsilon(1-\gamma)/2\gamma,$$

calculate:

$$P^*(s) = \max_{\mu}\left\{-C(\mu) + \gamma\sum_{s'\in S} T(s,\mu,s')V_t(s')\right\}$$

else, set $t = t + 1$ and go to Step 1.

D. Approximate Dynamic Programming

Approximate Dynamic Programming (ADP) is a collection of methods that are used to decrease the computational complexity of MDPs. Researchers from three different research communities have written dedicated books to this topic. From control systems theory, book by Bertsekas and Tsitsiklis [45] provides a fundamental theoretical foundation of the field. Specifically neural network approximations have been used in this text [45] to overcome the "curse of dimensionality" and the "curse of modeling". These curses have been bottlenecks to the practical application of stochastic control and dynamic programming to complex problems. From the perspective of artificial intelligence and computer science, book by Sutton and Barto [46] describe the field of ADP. Starting with intuitive examples and a definition of reinforcement learning, they present three fundamental approaches to reinforcement learning. The operational research (OR) perspective of ADP is presented by Powell [47]. Where the emphasis is on the high-dimensional problems that typically arise in OR. In [48], the authors present an algorithm that dynamically performs hierarchical decomposition of factored MDPs. Their algorithm is based on determination of causal relationship between states. Communication-based decomposition methods for decentralized MDPs are presented in [49]. A goal-based decomposition approach (similar to the approach adopted in this thesis) is presented in [50]. In [50], the decomposition is based on the additive terms in the reward function that correspond to different sub-goals. Decomposed MDPs are assigned sub-goals based on decomposition of the reward function. Optimal policies are computed for each sub-goal and finally merged together using a value function heuristic and best-first search to generate an approximate policy for the original task.

V. Incorporating Fault Tolerance in Spacecraft Mission Planning

Incorporation of fault tolerance in mission planning of spacecraft requires nontrivial innovation. On the other hand,

any of the methods discussed in the previous section can be employed for realization of fault tolerant mission planning. In this section we present some classic approaches for fault tolerant spacecraft motion control and processes control. We also identify the representation gaps in both types of control. As evident from discussion in section II, the models of processes and events differ fundamentally from the models of motion (kinematics and dynamics).

A number of algorithms for planning and scheduling as well as plan execution have been proposed in the Artificial Intelligence (AI) community [22][51][24][13]. A majority of these algorithms present the state as a set of discrete valued variables. This enables search-based algorithms to select, decompose, and sequence appropriate actions given the specified task-level goal and the observed system state. For spacecraft for which operations involve nontrivial uncertainty, reasoning is typically based on Bayesian and/or Markov Decision Process (MDP) models [52]. The MDP builds optimal policies that cater for the uncertainty involved in the state transitions of the spacecraft. Note that the MDP solves a slightly more general form of the AI planning problem where state transitions involve uncertainties and possess the Markov property. The AI literature includes many frameworks for spacecraft mission planning and execution, but only a few of them have been successfully deployed. This is mainly due to their computationally-intensive and often difficult-to-validate nature. The Remote Agent [22][53][54] offers a good focus for this paper due to its emphasis on fault tolerance for space applications. In the following subsection, we describe Remote Agent in more detail since it is one of the most successful multi-layer AI architectures implemented and deployed on a spacecraft. Alternatives to the AI based approaches include algorithms based on the theory of fault tolerant control [55][56][57]. Fault tolerant control uses physics based dynamical models with continuous time state variables as opposed to discrete valued variable in AI based approaches. On top of the physics based dynamical model, there is usually supervisory control logic for fault related decision making.

A. Artificial Intelligence Planning

Researchers from the Jet Propulsion Laboratory (JPL) and NASA Ames developed the Remote Agent (RA) AI architecture to enable autonomous onboard mission management [22][53]. Remote Agent has been tested on the Deep Space One (DS-1) spacecraft. RA is comprised of five components: 1) Planning Experts (PE), 2) Mission Manager (MM), 3) Planner and Scheduler (PS), 4) Smart Executive (EXEC), and 5) Mode Identification and Reconfiguration (MIR). Planning Experts (PE) are innovative software modules. The role of PE is to assist the Planner and Scheduler (PS) in two possible ways: the PE can either request new goals from PS or PE can compute planning solutions for PS. For example, in the navigation domain, PE may request updated engine thrust goals based on its estimation of the spacecraft orbit, whereas in the attitude domain PE might provide anticipated duration of specified turns and resulting resource consumption.

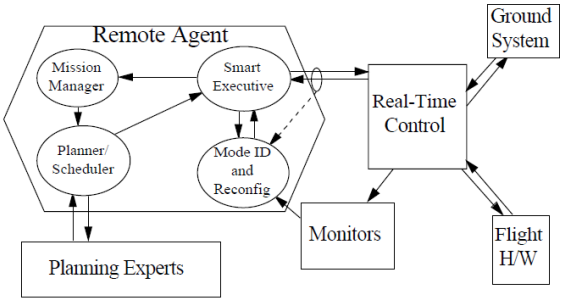

Fig. 3: The original Remote Agent architecture [22]

The role of the mission manager is to initiate PS activities based on long term mission objectives and current status of mission execution. The execution layer (EXEC) incorporates artificial intelligence and is responsible for main mission execution. EXEC also gathers data from onboard sensors and monitors to determine the mission execution status which is sent to the mission manager along with the request for new plans to be executed. Initial mission objectives are programmed before launch and later on new objectives can be uploaded from the ground station. MM works in phases or horizons of time. In a particular phase, the objectives to be achieved are combined with the current status of the spacecraft provided by EXEC. Then constraints are formulated based on the mission objectives (for current phase) and the spacecraft status to be sent to PS. The decomposition of mission objectives into short term phases allows for uninterrupted long term autonomy.

The role of the PS is to perform iterative repair based search [21] and chronological backtracking in order to calculate the set of tasks that extends the existing partial plan. PS consists of a plan database and Heuristic Scheduling Testbed System (HSTS) [51] that uses the plan database to calculate the consequences of each activity in the partial plan and to generate feasible sequences of activities that form a plan. Domain Description Language (DDL) [58] is used to specify domain constraints within HSTS. A finite set of symbolic state variables are used to describe the system state. Special variables "Tokens" are used to describe action and state literals. PS incorporates search-based planning and scheduling (in classical sense) that is implemented efficiently using parallel programming threads. Although the search mechanism in PS is classic, it is advanced to handle periodic and accumulative goals as well as the default goals.

EXEC is the plan execution system that is robust and multithreaded driven by events occurring inside the spacecraft. EXEC is a framework for customized control, fault diagnosis, and fault reconfiguration. It uses spacecraft state and current mission goals to determine activity to be performed. EXEC can handle interdependent activities and can request and execute plans with potentially uncertain outcomes and timings. EXEC determines primitive commands from the tasks involved in current mission plan based on the state of the spacecraft. In this way the planner can reason at higher level where EXEC handles low level execution. EXEC is also designed to compute fault responses during various activities. Execution Support Language (ESL) [20] is used in the development of EXEC which provides parallel execution, synchronization, error handling, and property locks [58]. A special submodule known as Mode Identification (MI) helps EXEC in monitoring of the

task execution and spacecraft state. When the given plan by MM is executed by EXEC, it provides MM with the current state of the spacecraft and requests new plan. In case of any fault rendering the spacecraft unable to execute the current plan, EXEC puts the spacecraft in safe mode and requests an alternate plan from MM. The alternate plan is generated without intervention from the ground station if within the capabilities of the MM. A certain level of robustness is achieved by EXEC through another special submodule called Mode Reconfiguration (MR) module. MR allows flexibility through deductive search and calculation of appropriate response to a given fault so that the MM may not have to be summoned.

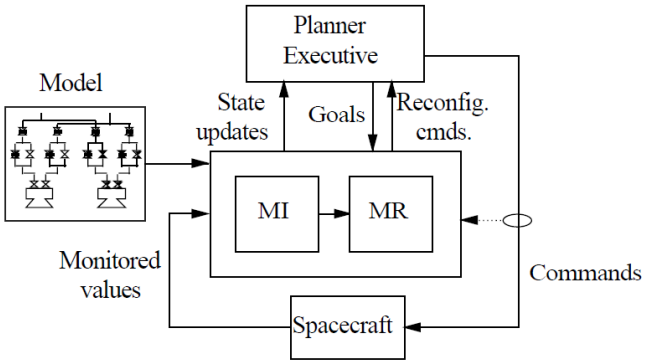

Fig. 4: Livingstone architecture [22]

The Mode Identification and Reconfiguration (MIR) capability in Remote Agent is provided by a special architecture known as the Livingstone [59]. Livingstone is a discrete model-based controller inserted between low level feedback control layer and high-level feed-forward reasoning in a physical system. MIR is responsible for calculation of reconfiguration of the systems (spacecraft) that mitigate or reduce the effect of faults on the mission goals. MIR has Mode Identification (MI) for processing sensor values into spacecraft mode of operation. Also Mode Reconfiguration (MR) in MIR computes the reconfiguration commands required to mitigate/reduce the effects of fault. The model of MIR is compositional, stochastic, and declarative with concurrency support. MI receives input data from EXEC and spacecraft sensors and combines it with the built-in model of the spacecraft to determine possible faults. Specifically, predicted sensor values from the model are compared to the actual sensor values to determine discrepancy. If a mismatch is found, then possible fault is searched for which best explains the mismatch. Fault recovery is handled by MR in coordination with EXEC. Whenever a fault occurs, MI informs EXEC and EXEC calls MR for possible actions. In call to MR, EXEC also conveys the mission goals and resource constraints so that only feasible reconfiguration is returned. MR solves the constraint satisfaction problem by performing reactive deduction and search with the help of unit propagation and propositional logic. Fig. 4 shows the architecture of Livingstone.

B. Control Systems Theory based Approach

The fault detection and reconfiguration discussed in the previous subsection is only at a discrete level. The model used is based on operational modes of the spacecraft and not the actual physical motion. Continuous time fault detection based on the model of the motion requires the use of the tools from control systems theory. Such tools make use of the knowledge of forces acting on the spacecraft and the resulting behavior. A lot of research has been done in this regard categorized in the literature as Fault Tolerant Control System (FTCS). FTCS typically is a three layered architecture [60][55][57]. The bottom layer consists of state estimation and control with additional capability of reconfiguration. Middle layer is usually a fault detection and diagnosis (FDD) [56] scheme. Top layer consists of a supervisory logic that manages the whole operation of the system by manipulating the lower layers in the light of information obtained from these layers and the high level system objectives. Fig. 5 shows example fault tolerant control system architecture. In this figure, **x** is the state vector for dynamic system, **u** is the control input vector, **y** is the output vector, **F** is the vector of fault flags, and *M* is a scalar indicating configuration mode of the reconfigurable controller.

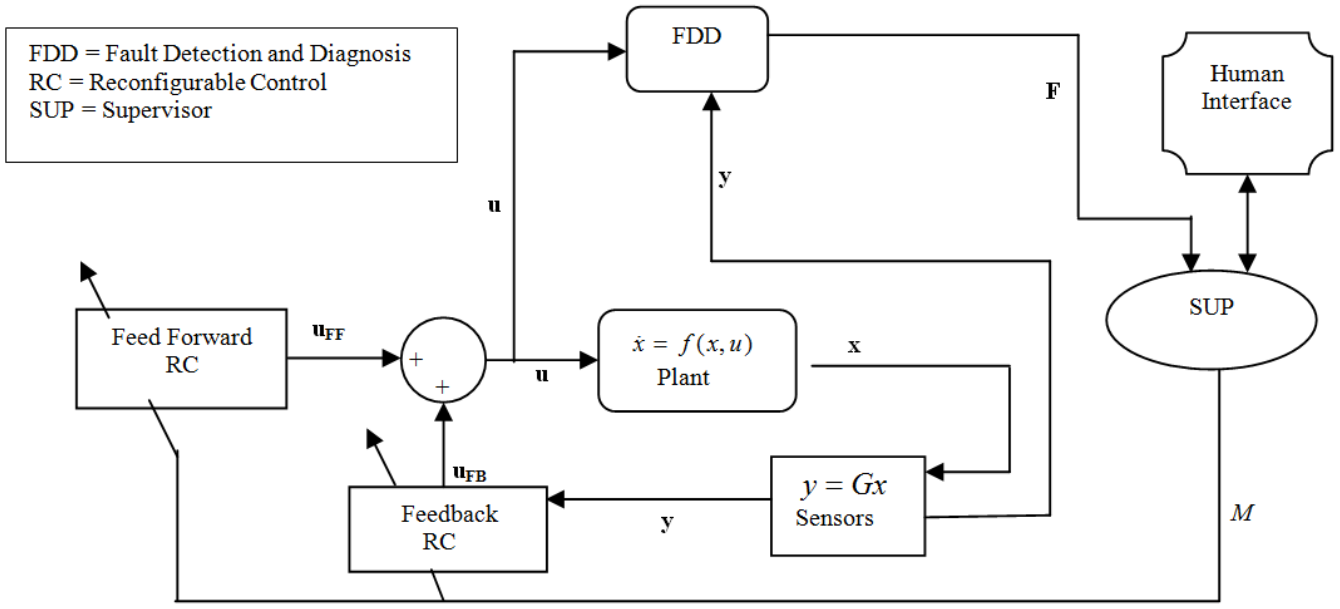

Fig. 5: Fault tolerant control architecture [56]

There are many formulations in terms of adaptation in control laws in FTCS. Specifically, there are two types of adaptation schemes i.e. active and passive [55]. The passive schemes rely on the robustness of the control systems and the design of the controller. Whereas the active schemes use online recalculation of the control law that is most appropriate given a certain fault condition. Sometimes active and passive schemes can be used in combination where passive serves as the baseline and active comes into play when inherent robustness is not enough to cater for the fault at hand. A fundamental tradeoff in fault tolerant control is between detection sensitivity and robustness. It is hard to detect a fault in a robust system whereas a non-robust system is susceptible to even small faults. For quick reconfiguration, active scheme of fault tolerance may incorporate pre-computed fault responses [57]. In general, FTCS can either mitigate the effect of a fault or reduce the same as much as possible [61].

As shown in Fig. 5, faults (**F**) are predicted by FDD layer of FTCS using the information of the input and output of the systems and built-in model of dynamics [62]. A signal known as residual signal indicates the deviation of the actual behavior (**x**) from the behavior depicted from the model of the dynamics ($f(\mathbf{x}, \mathbf{u})$). Different values of the residual correspond to different levels of a fault (or may refer to as different faults). In case of normal operation, all residual signals automatically turn out to be zero. Usually state estimation or parameter estimation or a mixture of both is used to implement FDD. There are two main types of FDD schemes i.e. data-based FDD approaches and model-based FDD approaches. Robustness is a key requirement for any FDD scheme. Specifically model be robust [63] otherwise a false alarm or a missed detection could occur rendering the FTCS to a completely devastating response. Many methods have been developed in order to incorporate robustness in FDD schemes e.g. averaging, statistical data processing, adaptive thresholds, and fuzzy decision-making [63][64]. A daunting challenge in the implementation of FTCS is to distinguish between disturbances in the system, the noise in the sensors, and the occurrence of faults. Methods have been developed in literature to decouple disturbance from the fault [56]. But the disturbance decoupling may sometimes lead to missed detection if some faults affect the system in the same manner as the disturbance does. For a complex system such as spacecraft, FTCS must be able to detect and diagnose coexisting multiple faults as well as incipient and abrupt faults.

The logic based supervisory layer is responsible for decisions of reconfiguration (M) based on its built-in reasoning algorithms and the information received from FDD. Some methods for implementation of supervisory logic are presented in [57][55][57], Some of the techniques are based on Fuzzy Logic, Intelligent Computing, Genetic algorithms, Neuro-Computing, and Probabilistic reasoning. Furthermore, supervisor logic may include failure mode effect analysis. Also state machines can be used to model various transitions.

C. Representational Gaps

So far two different types of fault tolerance architectures are discussed. One is MIR in remote agent and the other is FTCS in control systems community. While both architectures perform fault tolerance, the major difference is in the under lying model of the system (spacecraft) that is used for the purpose. The MIR uses discrete high level model of the spacecraft that includes operational modes and transitions between the modes. On the other hand FTCS uses the model of dynamics and kinematics of the spacecraft that is based on its physical motion. The sensor data required by MIR may include current, temperature, and voltage sensor data whereas FTCS may require data from magnetometers, gyroscopes, and accelerometers. Control commands in MIR may involve turning on or off various components and subsystems while commands in FTCS involve thruster firings, reaction wheel speed adjustment etc. MIR typically deals with various components and their interaction within the spacecraft while being unaware of the motion profile. On the other hand, FTCS deals with the motion profile while being unaware of the spacecraft component interactions. Such differences in reasoning and operation of MIR and FTCS are termed as *representational gaps* in these two architectures. It may be tempting to bridge the representational gaps by extending the models of either MIR or FTCS. Extending the analytical model used by MIR to include motion profile requires substantial increase in state space which may render the response of the MIR to anomalies be slow and sluggish. On the other hand incorporating inherently discrete component interaction dynamics into the continuous time motion profile model adds substantial complexities to the FTCS model. Therefore it is advisable to bridge the representational gaps by using both architectures and an interfacing mechanism [65] rather than making things complex by extending either of the modeling approaches.

D. Combining AI with Control Systems

The ultimate solution for fault tolerance in spacecraft mission planning is to use the combination of control systems and artificial intelligence. One way to achieve this goal is to use the famous three-tier architecture [20]. This architecture proposes the following layers in decision making 1) The controller, 2) The sequencer, 3) The deliberator. Controller layer uses the model from control systems theory and is able to handle the continuous time dynamics of the spacecraft. The sequencer uses the discrete AI model and makes high level decisions about mission-related activities. The deliberator is responsible for resolving conflicts and finding solutions for anomalous situations using complex computations and extensive search. Adoption of three-tier for fault tolerance in spacecraft mission planning is a less discussed area of research. The approach used in [65] is similar to a three-tier architecture where the focus is on the design of the deliberator layer. Also the deliberator layer has been implemented using an MDP which poses the issues of computational complexity. An approach for reduction in complexity is proposed in [65] but more rigorous treatment is required for generalization of the proposed method. Specific problem in designing a combined approach is to design a robust and reliable mechanism for highlighted blocks in Fig. **6**. With the existing knowledge base, it is possible to develop dynamics based and logic based fault diagnosis engines separately. But the real challenge is to make use of the combined information from these engines and reach a more reliable conclusion about the existence and nature of faults. According to maximum likelihood, combined information from a group of different sensors (or sources) is at least as reliable as the information

from the most reliable sensor among the group. A simple maximum likelihood based approach can be used for combining information from artificial intelligence based and dynamics based fault diagnosis. But such scheme would simply be an underutilization of the potential of information. There is more to obtain from series of data than just comparing numbers. For example machine learning can be used to identify trends and learn from statistics of faults and fault calls from each source. Incorporation of learning algorithms may increase the online computational overhead but is worth investigating. It is worth mentioning here that dynamics based fault diagnosis engines mostly cover faults in attitude estimation and control related equipment, e.g., attitude sensors and actuators. On the other hand, logic based model can cover faults in practically any component of the system. Therefore, when working on the combination of artificial intelligence and control systems, scope of faults is limited. Having said that, there might be components in the spacecraft which are not attitude sensors or actuators but their faults affect attitude control, e.g., circuit boards, wiring connectors, even the structure damage on the spacecraft can affect attitude control. Therefore it is important to combine artificial intelligence based diagnosis of whole system with dynamics based diagnosis of attitude control system in order to reach to the true cause of anomalous behavior of spacecraft.

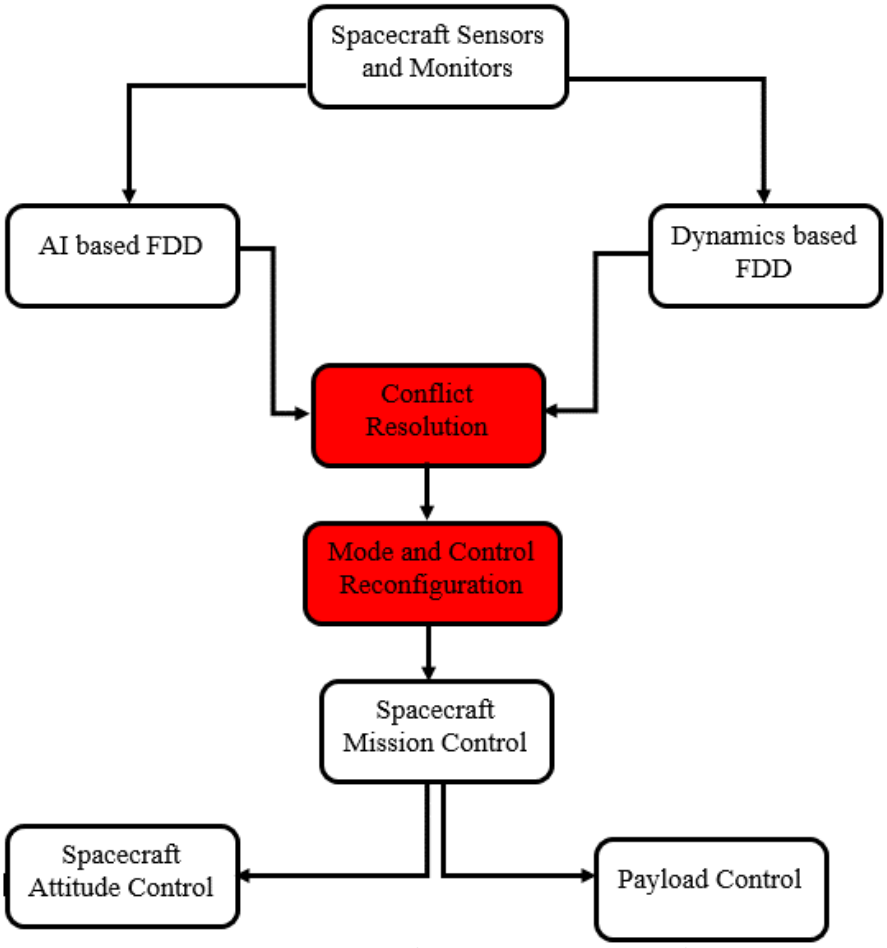

Fig. 6: Combined Approach for Fault Tolerance in Spacecraft

VI. Conclusion

Theories involved in spacecraft mission control and incorporation of fault tolerance have been reviewed. The paper provides a thorough overview of algorithms and key architectures from the AI and control systems literature. While there have been many approaches developed for incorporation of fault tolerance in spacecraft missions, a lot of questions remain unanswered. For example, what is the most efficient way of implementing fault tolerance? How can AI and control systems based models be fused in a way that achieves the pros of each approach while keeping complexity in check? How can a spacecraft anticipate faults and execute precautionary measures to avoid or manage failures should faults be experienced? The future of research in this regard is in the hybrid approach where methods from control systems theory are combined with those of AI to enable reasoning over the spectrum of discrete and continuous system properties required to model the spacecraft and its mission.

References

[1] S. Chien, M. Johnston, N. Policella, N., Frank, J et al "A generalized timeline representation, services, and interface for automating space mission operations" (*SpaceOps* 2012). Stockholm, Sweden. June 2012.

[2] S. Chien, R. Knight, A. Stechert, R. Sherwood, G. Rabideau, "Using iterative repair to increase the responsiveness of planning and scheduling for autonomous spacecraft". *International Joint Conference on Artificial Intelligence (IJCAI 1999)*. Stockholm, Sweden. August 1999.

[3] M. D. Johnson, and G. E. Miller, "Spike: intelligent scheduling of hubble space telescope observations", edited by M. Zweben and M. Fox, *Intelligent Scheduling*, Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[4] E. Atkins, E. Durfee, and K. Shin, "Planning and resource allocation for hard real-time, fault-tolerant plan execution." *Autonomous Agents and Multi-Agent Systems* 4.1-2 (2001): 57-78.

[5] A. Nasir, and E. Atkins, "Fault tolerance for spacecraft attitude management," AIAA *Guidance, Navigation, and Control Conference*, Toronto, Ontario, Aug. 2-5, 2010 (AIAA-2010-8301).

[6] G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative repair planning for spacecraft operations in the ASPEN system". *International Symposium on Artificial Intelligence Robotics and Automation* in Space (ISAIRAS 1999), Noordwijk, Netherlands. June 1999.

[7] J. R. Wertz, and W. J. Larson, *Space Mission Analysis and Design*. El Segundo, California: Microcosm Press, 2003.

[8] M. D. Shuster, "A survey of attitude representations". *Journal of Astronautical Sciences*, Vol 41, No 4, October-December, 1993, pp 439-517.

[9] P. Hughes, *Spacecraft Attitude Dynamics* Dover Publications, INC. New York copyright 1986, 2004.

[10] B. Wie, *Space Vehicle Dynamics and Control* AIAA Education Series 1998.

[11] H. Krishnan, N. H. McClamroch, and M. Reyhanoglu, "Attutude stabilization of a rigid spacecraft using two control torques: a nonlinear control approach based on the spacecraft attitude dynamics". *Automatica*, Vol. 30, No. 12, December, 1994, 1885-1897.

[12] E. J. Lefferts, F. L. Markley, M. D. Shuster, "Kalman filtering for spacecraft attitude estimation" *AIAA 20th Aerospace Science Meeting*, Orlando, Florida, January 11-14, 1982.

[13] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd Edition, Prentice-Hall, Upper Saddle River, New Jersey 07458, 2005.

[14] R. W. Conway, W. L. Maxwell, L. W. Miller, *Theory of Scheduling*, Dover Publications Inc., 31 East 2nd Street, Mineola, N.Y. 11501, 2003.

[15] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 4th edition, Springer Science+Business Media, LLC, 233, Spring Street, New York, NY, 10013, USA, 2012.

[16] M. Xiong, Q. Wang, and K. Ramamritham, "On earliest deadline first scheduling for temporal consistency maintenance," *Real-Time Systems*, 2008.

[17] Ł. Kruk, J. Lehoczky, S. Shreve, "Earliest-deadline-first service in heavy-traffic", *The annuals of applied probability*, 2004, Vol 14, No. 3, pp. 1306-1352.

[18] A. K. Atlas and A. Bestavros, "Statistical rate monotonic scheduling", *Proceedings of the 19th IEEE Real-Time Systems Symposium*, 1998, pp. 123-132.

[19] J. Zalewski, "What every engineer needs to know about rate monotonic scheduling: A Tutotial", *IEEE magazine*-95/1, IEEE Computer Society Press, 1995.

[20] E. Gat, "On three-layer architectures." *Artificial intelligence and mobile robots* 195 (1998): 210.

[21] N. Muscettola, B. Smith, S. Chien, C. Fry, G. Rabideau, K. Rajan, D. Yan, "On-board planning for autonomous spacecraft", in: *Proc. 4th International Symposium on Artificial Intelligence, Robotics. And Automation for Space* (i-SAIRAS), Tokyo, Japan, August 1997.

[22] N. Muscettola, P. P. Nayak, B. Pell, B. C. Williams, "Remote Agent: To boldly go where no AI system has gone before". *Artificial Intelligence*, 103(1-2):5--48, 1998.

[23] N. Muscettola, "HSTS: Integrating planning and scheduling". In Mark Fox and Monte Zweben, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

[24] R. Xu, P. Yuan-Cui, X. F. Xu, H. T. Cui "Multi-agent planning system for spacecraft". *Proceedings of the Second International Conference on Machine Learning and Cybernetics*, Xi'an, 2-5 November 2003, pp 1995-1999.

[25] S. Das, M. Sinha, K. D. Kumar "Reconfigurable magnetic attitude control of earth pointing satellites". *Proceedings of Institution of Mechanical Engineers*, Part G: Journal of Aerospace Engineering, 2010 224:1309. Published by SAGE. DOI: 10.1243/09544100JAERO681.

[26] M. Lovera, and A. Astolfi, "Spacecraft attitude control using magnetic actuators". *Automatica*, 40 (2004) 1405-1414. Copyright 2004 Elsevier Ltd.

[27] T. Pulecchi, M. Lovera, A. Varga, "Classical vs modern magnetic attitude control design: a case study". *GNC 2008 7th International ESA Conference*, Tralee, County Kerry, Ireland, 2008.

[28] E. Silani, and M. Lovera, "Magnetic spacecraft attitude control: a survey and some new results". *Control Engineering Practice*, 13 (2005) 357-371. Copyright Elsevier Ltd.

[29] P. Zentgraf, and D. Reggio, "Magnetic rate damping for satellites in LEO". *32nd Annual AAS Guidance and Control Conference*, Jan 31 – Feb 4, 2009, Breckenridge, Colorado.

[30] J. L. Crassidis, and F. L. Markley, "A minimum model error approach for attitude estimation". *AIAA Journal of Guidance Control and Dynamics*, 20 (6) (1997) 1241-1247.

[31] J. L. Crassidis, and F. L. Markley, "Unscented filtering for spacecraft attitude estimation". *AIAA Journal of Guidance Control and Dynamics*, 20 (4) (2003) 536-542.

[32] J. L. Crassidis, and F. L. Markley, "Attitude estimation using modified rodrigues parameters" *Proceedings of the Flight Mechanics/Estimation Theory Symposium*, (NASA/CP-1996-3333)NASA-Goddard Space Flight Center, Greenbelt, MD, 1996, pp. 71–83.

[33] J. L. Crassidis, F. L. Markley, Y. Cheng, "A survey of nonlinear attitude estimation methods". *AIAA Journal of Guidance, Control, and Dynamics*, 30 (1): 12-28, January 2007.

[34] R. M. Haralick, and G. L. Elliott, "Increasing tree search efficiency for constraint satisfaction problems". *Artificial Intelligence* 14 (1980) 263-313, Copyright © by North-Holland Publishing Company.

[35] R. Dechter, and J. Pearl, "Network based heuristics for constraint satisfaction problems". *Artificial Intelligence*, 34 (1) (1987), pp 1 – 38.

[36] R. Dechter, "Temporal constraint networks". *Artificial Intelligence*, 49 (1991) 61-95. Elsevier Science Publishers B.V.

[37] R. Dechter, *Constraint Processing*, Morgan Kaufmann Publishers an Imprint of Elsevier Science, San Francisco, California. Copyright 2003 by Elsevier Science (USA).

[38] S. C. Brailsford, C. N. Potts, B. M. Smith "Constraint satisfaction problems: algorithms and applications". *European Journal of Operational Research* 119 (1999) 557-581.

[39] G. Labinaz, M. M. Bayoumi, K. Rudie, "A survey of modeling and control of hybrid systems". *International Fedration of Automatic Control* 1997, published in Great Britan, S0066-4138(97)00019-0, Vol 21, pp. 79-92.

[40] B. Potocnik, G. Music, B. Zupancic "Model predictive control of discrete-time hybrid systems with discrete inputs". *ISA Transactions*, 44 (2005), 199-211.

[41] P. J. Antsaklis, "A brief introduction to the theory and applications of hybrid systems". *Proceedings of the IEEE, Special Issue on Hybrid Systems: Theory and Applications*, Vol. 88, No. 7, pp. 879-887, July 2000.

[42] S. Aberkane, J. C. Ponsart, M. Rodrigues, and D. Sauter, "Output feedback control of a class of stochastic hybrid systems", *Automatica*, Elsevier 2008.

[43] P. R. Kumar, and P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control*, Prentice Hall Inc., Englewood Cliffs, New Jersey 07632, 1986.

[44] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, © John Wiley and Sons Inc. (1994).

[45] D. Bertsekas, and J. Tsitsiklis, *Neuro-dynamic programming*, Athena Scientific, Belmont, MA, 1996.

[46] R. Sutton, and A. Barto, *Reinforcement learning*, The MIT Press, Cambridge, Massachusetts, 1998.

[47] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York: Wiley, 2007.

[48] A. Jonsson, and A. Barto, "A causal approach to hierarchical decomposition of factored MDPs", *Proceedings of the Twenty Second International Conference on Machine Learning* (ICML 05), 2005.

[49] C. V. Goldman, and S. Zilberstein, "Communication-based decomposition mechanisms for decentralized MDPs", *Journal of Artificial Intelligence Research*, 32 (2008) 169-202.

[50] C. Boutilier, R. I. Brafman, C. Geib, "Prioritized goal decomposition of Markov decision processes: toward a synthesis of classical and decision theoretic planning", *Proc. 15th Intl. Joint Conf. on AI* (IJCAI-97), Nagoya, August, 1997.

[51] M. Zweben, and S. Mark, *Fox Intelligent Scheduling* (Chapter 6), Morgan Kaufman Publishers, San Francisco, California, 1994.

[52] C. Boutilier, T. Dean, S. Hanks, "Decision-theoretic planning: structural assumptions and computational leverage" *Journal of Artificial Intelligence Research* 11 (1999) 1-94.

[53] K. Havelund, M. Lowry, S. Park, C. Pecheur, J. Penix, W. Visser, and J. White, "Formal analysis of the remote agent before and after flight", *In Proceedings of the 5th NASA Langley Formal Methods Workshop*, June 2000.

[54] B. Pell, et al "A hybrid procedural/deductive executive for autonomous spacecraft". *Autonomous Agents and Multi Agent Systems*, 2, 7-22(1999) Kluwer Academic Publishers, 1999.

[55] R. J. Patton, "Fault-tolerant control: The 1997 situation", *In Proceedings of the 3rd IFAC symposium on fault detection, supervision and safety for technical processes*, (pp. 1033–1055), August, 1997.
[56] Y. Zhang, J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual Reviews in Control*, Elsevier Ltd, Volume 32, Issue 2, December 2008, Pages 229-252.
[57] M. Blanke, R. Izadi-Zamanabadi, R. Bogh, and Z. P. Lunau, "Fault tolerant control systems—A holistic view", *Control Engineering Practice*, 5(5), 693–702, 1997.
[58] T. L. Dean, D. V. McDermott, "Temporal data base management", *Artificial Intelligence* 32 (1987) l-55.
[59] C. B. Williams, and P. P. Nayak, "A model-based approach to reactive self-configuring systems," *in Proceedings of AAAI-96*, pages 971-978, AAAI, AAAI Press, Cambridge, Mass., 1996.
[60] D. H. Zhou, and P. M. Frank, "Fault diagnostics and fault tolerant control". *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, No. 2, April 1998.
[61] Y. M. Zhang, and J. Jiang, "Fault tolerant control system design with explicit consideration of performance degradation", *IEEE Transactions on Aerospace and Elctronic Systems*, 2003, 37 (3).
[62] A. Chamseddine, H. Noura, M. Ouladsine, "Sensor fault detection, identification and fault tolerant control: application to active suspension". *American Control Conference*, 2006 2006 Jun 14 (pp. 6-pp). IEEE.
[63] Y. M. Zhang, and J. Jiang, "Integrated active fault tolerant control using IMM aproach". *IEEE Transactions on Aerospace and Electronic Systems*, IEEE, Vol. 37, No. 4, 1221-1235, October 2001.
[64] L. W. Ho, and G. G. Yen, "Reconfigurable control system design for fault diagnosis and accommodation", *International Journal of Neural Systems*, 12(6), 2002, pp 497–520.
[65] A. Nasir, E. M. Atkins, and I. V. Kolmanovsky, "Conflict resolution algorithms for fault detection and diagnosis" *AIAA Infotech@Aerospace Conference*, St. Louis, Missouri, March. 29-31, 2011 (AIAA-2011-1587).

Ali Nasir received the B.Sc. degree in electrical engineering from the University of Engineering and Technology Taxila, Taxila, Pakistan, in 2005, the first M.Sc. degree in electrical engineering, the second M.Sc. degree in aerospace engineering, and the Ph.D. degree in aerospace engineering from the University of Michigan, Ann Arbor, MI, USA, in 2008, 2011, and 2012, respectively. He is currently the Head of Electrical Engineering Department, University of Central Punjab, Lahore, Pakistan. His current research interests include approximate dynamic programming, Markov decision processes, fault tolerant control, and optimal control for multiagent systems. Dr. Nasir was a recipient of the Fulbright Scholarship in 2007.

Ella M. Atkins received the B.S. and M.S. degrees in aeronautics and astronautics from the Massachusetts Institute of Technology, Cambridge, MA, USA, and the M.S. and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor, MI, USA. She served on the Aerospace Engineering Faculty with the University of Maryland College Park, College Park, MD, USA. She is a Professor with the Department of Aerospace Engineering, University of Michigan, where she is the Director of the Autonomous Aerospace Systems Laboratory and the Associate Director of Graduate Programs for the Robotics Institute. She has established a long-term research program in decision-making and control to assure safe contingency management in manned, and unmanned aircraft applications. Dr. Atkins is a past Chair of the AIAA Intelligent Systems Technical Committee, AIAA Associate Fellow, small Public Airport Owner/Operator (Shamrock Field, Brooklyn, MI, USA), a Private Pilot (Aircraft Single Engine Land), and holds a Part 107 UAS certificate. She served on the National Academy's Aeronautics and Space Engineering Board from 2011 to 2015, the Institute for Defense Analysis Defense Science Studies Group, from 2012 to 2013, and an NRC committee to develop an autonomy research agenda for civil aviation, from 2013 to 2014.

Ilya Kolmanovsky received the M.S. and Ph.D. degrees in aerospace engineering, and the M.A. degree in mathematics from the University of Michigan, Ann Arbor, MI, USA, in 1993, 1995, and 1995, respectively. He was with Ford Research and Advanced Engineering, Dearborn, MI, USA, for close to 15 years. He is currently a Professor with the Department of Aerospace Engineering, University of Michigan. His current research interests include control of systems with constraints, control of automotive and aerospace propulsion systems, and spacecraft control applications. In 2011, he was a Summer Faculty Fellow with the Space Vehicles Directorate of Air Force Research Laboratory, Albuquerque, NM, USA. He has co-authored over 250 journal and conference articles and is named as an inventor on 84 U.S. patents and derivative international patents. Dr. Kolmanovsky was a recipient of the Donald P. Eckman Award of American Automatic Control Council, the IEEE TRANSACTIONS ON CONTROL SYSTEMS Technology Outstanding Paper Award, and the several Ford Motor Company Technical Achievement, Innovation, and Publication awards.