Robust Supervisory Control of a Spacecraft Propulsion System

Farid Yari, Shahin Hashtrudi-Zad*, and Siamak Tafazoli

In this paper the theory of supervisory control of discrete-event systems is used to develop command sequences for turning on and off a spacecraft propulsion subsystem. The subsystem considered is a simplified version of the Propulsion Module Subsystem of the Cassini spacecraft. The supervisor controls the system in such a way that the design specifications are satisfied in both normal and faulty modes of operation. The study shows that to meet the specifications of both modes, the supervisor has to be a “robust” supervisor, and that a conventional (non-robust) supervisor could lead to engine getting stuck in shutdown state.

Index Terms—Supervisory control, Spacecraft propulsion, Fault recovery, Discrete-event systems, Robust control.

I. INTRODUCTION

This paper is concerned with the design of supervisory control system for spacecraft propulsion system. Specifically we seek to design a controller (supervisor) that issues appropriate command sequences to turn on and off the engines of a simplified spacecraft propulsion system. The objective is to enhance spacecraft autonomy. The subject of spacecraft autonomy has been studied by many researchers. Notably, the diagnosis and recovery system, Livingstone, has been designed based on probabilistic discrete models [Muscettola et al. (1998)]. Livingstone was tested on Deep Space 1 (1998) and its successor, Livingstone 2, was used on Earth Observing 1 (2000). In another approach, the methods of formal verification have been used to verify if a given control logic satisfies the design specifications (see, e.g. [Pekala et al. (2008)] and [Bensalem et al. (2010)]).

In this paper we apply the Ramadge-Wonham theory of supervisory control ([Ramadge and Wonham (1987); Wonham (2014)]) to systematically design controller (supervisor) from the design specifications based on deterministic discrete-event models. The resulting supervisor (which will be in the form of an automaton) is guaranteed to satisfy the specifications. We study a simplified version of the Propulsion Module Subsystem (PMS) of the Cassini spacecraft. Our study shows that the application of conventional (non-robust) supervisory control [Ramadge and Wonham (1987)] does not result in the appropriate command sequences. However, an extension of this approach to the Robust Supervisory Control ([Saboori and Hashtrudi Zad (2006)]) produces the desired solution.

The concept of robust control arises in control theory in dealing with modeling uncertainties or model changes. Several approaches have been explored for robust supervisory control of discrete-event systems (DES). In [Lin (1993)], the plant model belongs to a finite family of DES models $G_1,\ldots,G_n$, and the objective is to design a supervisor such that all plant models under supervision satisfy a common design specification. [Bourdon et al. (2005)] extends the results of [Lin (1993)] to the case involving separate design specifications for each plant model and considers the nonblocking property. The nonblocking property is guaranteed through the (sufficient condition of) nonconflicting property. [Saboori and Hashtrudi Zad (2006)] extends the results of [Bourdon et al. (2005)] to the case of control under partial observation. Furthermore, [Saboori and Hashtrudi Zad (2006)] replaces the nonconflicting property with the $G_i$-nonblocking property to obtain a set of necessary and sufficient conditions for the solvability of the robust control problem.

Robust control is used to deal with model changes. For instance, in some fault recovery problems, the plant starts in normal mode and may enter one of several faulty modes. Each mode can have its own design specification and set of marked states. This control and fault recovery problem can be solved as a robust control problem [Saboori and Hashtrudi Zad (2005)].

[Yari and Hashtrudi Zad (2016)] develops automaton-based computational procedures for the design of robust supervisors of [Saboori and Hashtrudi Zad (2006)]. The procedures have been implemented in MATLAB environment using Discrete Event Control Kit (DECK) ([DECK (2013)]). In this paper, these procedures are used to solve the control and fault recovery problem for the simplified spacecraft propulsion system. Our study shows that to arrive at an appropriate control logic satisfying all design specifications, the supervisor has to be a robust supervisor. Furthermore, the features of the system studied here that necessitate the use of robust supervisor seem to be present in many other aerospace systems as well.

This paper is organized as follows. Sec. II reviews some preliminaries of supervisory control. Sec. III formulates the supervisory control and fault recovery problem. Sec. IV discusses the simplified propulsion system and the design of supervisor. Sec. V further analyzes the results and provides comments on the methodology and its relation to other approaches. The paper is concluded in Sec. VI.
II. Preliminaries

In this section, we briefly review the supervisory control of DES [Wonham (2014); Cassandras and Lafortune (2008)].

Languages and Automata. A finite set of symbols, Σ, is called an alphabet. A sequence of symbols σ₁σ₂...σₙ (with σᵢ ∈ Σ) is called a word or sequence. A sequence with no symbols is called the empty sequence and denoted by ε, Σ* denotes the set of all finite sequences. Any subset of Σ* is called a language. A deterministic automaton $G = (X, Σ, η, x₀, X_m)$ is a model for a discrete-event system with the state set, $Σ$ the finite set of events, $η : X × Σ → X$ the partial transition function, $x₀$ the initial state and $X_m ⊆ X$ the set of marked states. The set of sequences that can be generated by automaton $G$ (from initial state $x₀$) is called the closed behavior of $G$: $L(G) = \{ s ∈ Σ* | η(x₀, s) is defined \}$.

The subset of $L(G)$ consisting of sequences that end in a marked state is referred to as the marked behavior of $G$: $L_m(G) = \{ s ∈ L(G) | η(x₀, s) ∈ X_m \}$.

A state $x$ in $X$ is called reachable if there exists a sequence $s ∈ L(G)$ from $x₀$ to $x$. Also, a state $x$ is called coreachable if there is a sequence, say $s$, from $x₀$ to some marked state. A DES $G$ is nonblocking if every reachable state of $G$ is coreachable. A nonblocking automaton is free of deadlocks and livelocks. An automaton is trim if all of its states are reachable and coreachable. The product of automata $G_1$ and $G_2$ is denoted by $G_1 × G_2$ and generates $L(G_1) ∩ L(G_2)$ and marks $L_m(G_1) ∩ L_m(G_2)$. The synchronous product (parallel composition) of $G_1$ and $G_2$ is denoted by $sync(G_1, G_2)$ and can be used to model the joint operation of $G_1$ and $G_2$.

Supervisory Control. Consider a plant modeled by a DES $G$. Suppose a language $E ⊆ L_m(G)$ represents the legal marked behavior; i.e., the marked sequences that satisfy the design specifications. In the supervisory control theory, a supervisor $S$ is to be designed to restrict the behavior of the plant $G$ to the legal behavior $E$ and to ensure that the resulting system is nonblocking (Fig. 1). It is assumed that the plant event set $Σ$ can be partitioned into two disjoint subsets, $Σ_c$ the set of controllable events, and $Σ_{uc}$ the set of uncontrollable events. The supervisor is allowed to disable the controllable events only. A supervisory control for DES $G$ can be designed as a map $S : Σ* → Γ_Σ$ where $Γ_Σ = \{ γ ∈ P libr(Σ) | Σ_{uc} ⊆ γ \}$ denotes the set of all control patterns in $Σ$. In this context, for $s ∈ Σ*$ (generated in $G$), $S(s)$ is the set of events enabled by the supervisor. Let $S/G$ represent the plant $G$ under the supervision of $S$ (or the closed-loop system). Thus the Supervisory Control Problem (SCP) is to find a supervisor $S$ such that (1) $L_m(S/G) ⊆ E$, and (2) $S/G$ is nonblocking. ((1) and (2) imply that all sequences in $L(S/G)$ are legal.)

Next we consider the robust supervisory control problem which can be regarded as an extension of SCP.

Problem 1: Robust Nonblocking Supervisory Control Problem (RNSCP) [Bourdon et al. (2005); Saboori and Hashtrudi Zad (2006)]. Consider the set of plant models $G = \{ G_1, ..., G_n \}$ with $G_i = (X_i, Σ_i, η_i, x₀,i, X_{m,i})$ where $i ∈ I = \{ 1, 2, ..., n \}$. Let the controllable and uncontrollable sets of events in $G_i$ be denoted by $Σ_{c,i}$ and $Σ_{uc,i}$ respectively. It is assumed that all plant models agree on the controllability of events, that is $Σ_{c,i} ∩ Σ_{uc,i} = ∅$ for all $i, j ∈ I$. Furthermore, suppose language $E_i$ denotes the design specification for plant model $G_i$ and hence $E_i = K_i ∩ L_m(G_i)$ denotes the corresponding legal marked behavior. Design a supervisor $S$ such that (1) $L_m(S/G_i) ⊆ E_i$ and (2) $S/G_i$ is nonblocking, for all $i ∈ I$.

Thus in RNSCP we seek to find a supervisor $S$ such that (for every $i$) $S/G_i$ is nonblocking and satisfies its design specification.

Let $Σ = \bigcup_{i ∈ I} Σ_i$ and define $E$ as

$$E = \bigcap_{i ∈ I} (E_i \cup (Σ* − L_m(G_i))) \cap \bigcup_{i ∈ I} L_m(G_i) \tag{1}$$

The solutions of RNSCP can be characterized in terms of relative-closed, controllable and $G$-nonblocking sublanguages of $E$ [Saboori and Hashtrudi Zad (2006)]. Let $RCNb(E, G)$ denote the set of relative-closed, controllable and $G_i$-nonblocking sublanguages of $E$ (for all $G_i ∈ G$). This set is nonempty and has a supremal element denoted by $E^↓ = Sup(RCNb(E, G))$. Then $E^↓$ is the maximally permissive solution of RNSCP. A trim automaton $S$ that marks $E^↓$ can be regarded as an implementation of a maximally permissive supervisor with $S × G_i$ modeling the system under supervision $S/G_i$. [Yari and Hashtrudi Zad (2016)] provides a computational procedure to find an automaton to mark $E^↓$. This algorithm has been implemented in MATLAB environment using DECK [DECK (2013)].

III. Supervisory Control and Fault Recovery Problem

In this section we explain how a problem of control and fault recovery can be cast as a robust control problem. Consider a DES plant $G$ that starts in normal mode $N$. The plant may enter a set of permanent failure modes. For simplicity we assume a single failure mode $F$ (Fig. 2). Let $G_N$ denote the subautomaton of $G$ with $X_N$ as its state set representing the normal mode, and $G_{NF}$ represent the behavior of the system in normal ($N$) and faulty ($F$) modes. In this case, since we have assumed one failure mode, $G_{NF}$ and $G$ are the...
same. Furthermore, suppose in each mode we have multiple sets of marked states, say two, and denote them by $X_{m,1}^N$ and $X_{m,2}^N$ (normal mode) and $X_{m,1}^F$ and $X_{m,2}^F$ (faulty mode). $X_{m,1}^N$ and $X_{m,2}^N$ could for example represent the “off” states and “operational” states in normal mode. Similarly, $X_{m,1}^F$ and $X_{m,2}^F$ could represent “off” states and “operational” states in faulty mode. We use $G_{N,1}$ and $G_{N,2}$ to represent $G_N$ with $X_{m,1}^N$ and $X_{m,2}^N$ as their marked states. Automata $G_{NF,1}$ and $G_{NF,2}$ similarly are copies of $G_{NF}$ with $X_{m,1}^F$ and $X_{m,2}^F$ as marked states.

The control and fault recovery problem is to find a supervisor $S$ such that the system under supervision remains nonblocking in normal and faulty modes and satisfies the design specifications of each mode:

$$L_m(S/G_{N,i}) \subseteq E_{N,i} \quad i = 1, 2$$

$S/G_{N,i}$ is nonblocking

$$L_m(S/G_{NF,i}) \subseteq E_{NF,i} \quad i = 1, 2$$

$S/G_{NF,i}$ is nonblocking

($E_{N,i}$ and $E_{NF,i}$ ($i = 1, 2$) are the legal behaviors.)

The above problem can be solved as a robust nonblocking supervisory control problem for the family of models $S = \{G_{N,1}, G_{N,2}, G_{NF,1}, G_{NF,2}\}$.

Next we turn our attention to the simplified spacecraft propulsion system.

IV. CONTROL AND FAULT RECOVERY IN SPACECRAFT PROPULSION SYSTEM

In this paper we study supervisory control of a simplified version of the Propulsion Module Subsystem (PMS) of the Cassini spacecraft [Leeds et al. (1996); Morgan (2010)]. Fig. 3 shows the simplified propulsion system which consists of two propellant tanks and two engines E1 and E2. The valve assembly for E1 includes valves $V_1$, $V_2$, normally-open pyro valves $PV_1$ and $PV_2$, and normally-closed pyro valves $PV_3$ and $PV_4$. Two pressure sensors measure pressures $P_1$ and $P_2$, and a temperature sensor $T_1$ monitors chemical reactions and thrust generation in E1. The valve assembly for E2 is simpler consisting of valves $V_3$, $V_4$ and pressure sensors $P_3$, $P_4$ and temperature sensor $T_2$. When the fuel paths between the propellant tanks and engines are open, propellants can combine, ignite and produce thrust. In this example, we intend to design a supervisor that in response to high-level Master Controller commands, fires the engine in such a way that the design specifications (to be discussed later) are satisfied.

**Plant DES Model**

In our problem, for simplicity, only valve $V_1$ is assumed prone to “stuck-open” failure and the other valves are assumed fault-free. Fig. 4 shows the DES model of valve $V_1$. (Marked states are shown with double circles.) We assume that the valves are initially closed. The “stuck-open” failure mode of $V_1$ is permanent (state “SO”) and the valve never returns to normal mode. The models of $V_2$, $V_3$ and $V_4$ are similar but without the failure event and faulty state (SO). The models of the normally-open pyro valves $PV_1$ and $PV_2$ are shown in Fig. 5. These valves can be closed only once. The models of normally-closed valves $PV_3$ and $PV_4$ are similar but their initial states is “C” and their event $PV_i - O$. The models of the four pressure sensors and two sensors measuring temperature as well as the Master Controller are shown in Fig. 6. Start and stop commands can be generated at any time by the Master Controller (Fig. 6(c)). Valve events $V_i - O$, $V_i - C$, $PV_i - O$ and $PV_i - C$ are controllable. Failure event $V_1 - SO$, sensor events ($P_i H$, $P_i L$, $T_i H$ and $T_i L$) and Master Controller events (stop and start) are uncontrollable.

**Fig. 3. Simplified spacecraft propulsion module subsystem.**

**Fig. 4. DES model of valve $V_1$.**

**Fig. 5. Normally-open pyro valves $PV_1$ and $PV_2$.**

**Fig. 6. Sensors and Master Controller.**
To complete the model, we build DES models INT1 to INT4 to describe the effect of valve positions on the pressure sensors, and INT5 and INT6 to model the reading of the temperature (thrust) sensors as a function of the pressures measured by \( P_1, P_2, P_3 \) and \( P_4 \). The reading of pressure sensor \( P_1 \) depends on the state of \( V_1, PV_1 \) and \( PV_3 \). Specifically the pressure goes high only when \( PV_1 \) is open, and either \( V_1 \) is open or stuck-open. \( P_3 \) selfloops are added to other states. INT2, INT3 and INT4 are constructed similarly for pressure sensors \( P_2, P_3 \) and \( P_4 \). Now we proceed to model the interactions between pressure sensor readings and the thrust of engines. The thrust of an engine goes high only when the corresponding pressure sensors show high pressure reading. Fig. 7 shows these interactions for engine E1. INT6 (for engine E2) is built similarly.

The plant model \( G \) is obtained by the synchronous product of all component models \((V_1, ..., V_4, PV_1, ..., PV_4, P_1, ..., P_4, T_1, T_2)\) and interaction models \((\text{INT1}, ..., \text{INT6})\). The resulting plant model has 16384 states and 278528 transitions.

![Fig. 7. INT5: Interactions between pressure sensors and temperature sensor for engine E1.](image)

**Design Specifications**

Broadly speaking, the design specifications require the use of engine E1 for normal mode and E2 in case of valve \( V_1 \) failure and prohibit simultaneous firing of E1 and E2. In normal (resp. faulty) mode, the supervisor must be able to turn on E1 (resp. E2) and turn off E1 (resp. E2). (Other requirements that deal with other issues such as fuel waste are not considered for brevity.) The detailed design specifications are as follows. In **normal mode**: (a) The system should wait for the start command to start firing engine E1; (b) After the start command is issued by the master controller, any start and stop command should be ignored during the start-up procedure until thrust is generated; (c) When the thrust is high, once the master controller issues the stop command, the shutdown procedure starts; (d) During the shutdown procedure, any start and stop command must be ignored. The automaton \( SP_1 \) that models the start-up and shutdown procedure in normal mode according to the design specifications is shown in Fig. 8.

Engine 2 is a backup engine and must be used in case of engine 1 failure. When fault (i.e. valve \( V_1 \) stuck-open) occurs, engine 1 shall no longer be used and the system must switch over to engine 2 based on the current state of the system. The design specifications for the **faulty mode** are: (a) If the fault occurs before a start command, the system should switch completely to engine 2 and wait for the start command. In other words, the procedure for starting and shutting down engine 2 should be followed; (b) If the fault occurs after the master controller issues a start command, engine 1 should continue firing until the master controller issues the stop command and engine E1 is turned off. Then the system must switch to engine E2 for future maneuvers; (c) If the fault occurs after the master controller issues a stop command but before engine E1 is turned off, the engine must be turned off and then the system should switch over to engine E2.

The DES model of these specifications is shown in Fig. 11. **Robust Supervisory Control**

This supervisory control problem is a problem of fault recovery with a normal and a faulty mode. In each mode there are two sets of marked states, one for thrust low (engine off) and another for thrust high (engine on). The problem can be solved as a robust control problem with four plant models \( G_{N,off}, G_{N,on}, G_{NF,off}, G_{NF,on} \) (resp. \( G_{N,off} \)) is the subautomaton of plant model \( G \) with states in normal mode \( G_N \) with thrust low (resp. high) states marked. This marking can be done by the synchronous product of \( G_N \) with automaton Marker N in Fig. 9 with state 1 (resp. state 2) of Marker N marked. Similarly \( G_{NF,off} \) (resp. \( G_{NF,on} \)) are obtained by the synchronous product of \( G \) with Marker F (Fig. 10) with state 3 (resp. state 4) marked.

![Fig. 8. DES model for start-up and shutdown procedure in normal mode according to design specifications.](image)

![Fig. 9. Mark state 1 for OFF and state 2 for ON.](image)

Automata for legal marked behaviors are obtained as

\[
E_{N,1} = SP_1 \times G_{N,off}, \quad E_{N,2} = SP_1 \times G_{N,on}
\]

\[
E_{NF,1} = SP_2 \times G_{NF,off}, \quad E_{NF,2} = SP_2 \times G_{NF,on}
\]

The solution of robust control problem (RNSCP) is obtained using the computer code. The solution is an automaton with

2To each state of \( SP_1 \) selfloops for all events except “start”, “stop”, “TILH” and “TILH” are added. The selfloops are for events that are not restricted by design specification \( SP_1 \).
Fig. 10. Mark state 3 for OFF and state 4 for ON.

944 states and 11438 transitions. This automaton marks $E^\uparrow$ and is a model of the supervisor.

To illustrate what the robust supervisor does, consider a sample sequence from the system under supervision depicted in Fig. 12. Master Controller issues a start command. After valves $V_1$ and $V_2$ become open, pressures $P_1$ and $P_2$ become high. At this stage the thrust of $E_1$ becomes high and the first engine fires. Next valve $V_1$ becomes stuck-open (It is assumed the failure is diagnosed quickly and hence is treated an observable event.) Next when a stop command is issued, engine $E_1$ is switched off. This is done by firing $PV_1$ to shut the path between the oxidizer tank and engine $E_1$ and by closing valve $V_2$ which shuts off supply from the fuel tank. Following another start command from the Master Controller, the system switches to engine $E_2$ and the start-up procedure for $E_2$ opens valves $V_3$ and $V_4$, the pressure sensors $P_3$ and $P_4$ show high pressure and $E_2$ fires. Later, when a stop command comes from Master Controller, engine $E_2$ is turned off by closing the corresponding valves $V_3$ and $V_4$. When Master Controller issues another start command to generate thrust, engine $E_2$ is being used and the same sequence happens to complete a thrust on and thrust off sequence.

Remark 1: It should be noted that it is necessary to ensure nonblocking with respect to multiple sets of marked states (and not just one set). Suppose we choose the $E_1$ off states as the only marked state set for normal mode. Assume $E_1$ is “on” in normal mode and Master Controller issues thrust shutoff command. The conventional (non-robust) supervisory control solution allows the system to close $PV_1$ to turn off the thrust of engine 1. But the supervisor will no longer be able to turn on $E_1$ since the pyro valve cannot be opened again. The robust supervisor designed based on RNSCP however does not fire $PV_1$ and $PV_2$ unless a fault occurs.

V. DISCUSSION

In this section, we discuss some issues related to the problem studied here and the approach used to solve it.

Robustness: As mentioned in Remark 1 in the previous section, to solve the control and fault recovery problem for the propulsion system, the robust supervisory control problem, RNCSP, should be considered and that a non-robust supervisor does not yield the desired solution. There are two problem and plant features that are the reasons for this result:

- First we want nonblocking property with respect to more than one set of marked states. Also, the plant model changes as a result of fault.
- Second, in the propulsion system studied, there were irreversible activities; in particular, the opening (resp.

In the configuration shown in Fig. 3, pyro valves $PV_3$ and $PV_4$ are redundant. In fact, it can be validated that the RNSCP still has a solution even if $PV_3$ and $PV_4$ are removed from the propulsion system. This shows that the RNSCP and its maximally permissive solution can be used as a design tool to validate different valve configurations. Finally, it should be noted that the purpose of having $PV_3$ and $PV_4$ in Fig. 3 is to have the ability for handling stuck-closed failures of valves $V_1$ and $V_2$ (which have not been studied in this example).
closing) of normally-closed (resp. normally-open) pyro valves and the permanent failure event are irreversible. These two features are present in many aerospace systems that have to be able to accommodate permanent faults with limited onboard resources (such as pyro valves and non-resettable fuses). In such cases, a robust supervisor must be used instead of a conventional (non-robust) supervisor. It should be noted that the above two features may not be present in other categories of systems. For example, for manufacturing systems reversibility may be a very desirable property.

**Plant Model:** In this paper we limited our attention to a single failure mode. This has been primarily due to the execution time of our computer code in MATLAB environment. With a compiled code, say in C, we would be able to study more complex problems. Furthermore, other theoretical developments in the theory of supervisory control, for instance, modular and decentralized control, and symbolic calculations allow much larger problems to be tackled (see, e.g. [Brandin and Charbonnier (1994)] and [Ma and Wonham (2006)]).

The DES models used in this paper are untimed and assume full event observation. Robust control for timed DES models is discussed in [Bourdon et al. (2005)] and for partial event observation in [Saboori and Hashtrudi Zad (2006)]. We have developed computational algorithms for the case of partial event observation and will publish them in the future.

**Related Work:** [Muscettola et al. (1998)] provides an overview of the model-based diagnosis and reconfiguration system, Livingstone. Livingstone uses probabilistic discrete models. Probabilistic models are used to focus on the more likely scenarios in order to reduce computational complexity. The result is a system that can work in real time. Livingstone was implemented on Deep Space 1. The drawback is that every now and then, the actual situation is not among the likely scenarios. Our work is based on deterministic models and can be regarded as a step in the direction of considering all possible scenarios.

[Pekala et al. (2008)] and [Bensalem et al. (2010)] use model checking tools to verify the design of control logic. In [Bensalem et al. (2010)], for instance, the plant is modeled with discrete models in BIP framework. Next based on design specifications, the supervisor modules, called connectors, are represented in BIP framework. Then the safety and deadlock-freedom properties are verified using model checker D-finder. D-finder and similar tools, use symbolic calculations to handle problems far more complex than the one considered in this paper. The main difference between the approach used in this paper and that in [Pekala et al. (2008)] and [Bensalem et al. (2010)] is that here we start from the design specifications and systematically design a control logic (supervisor) that satisfies the specifications. [Pekala et al. (2008)] and [Bensalem et al. (2010)] however start with given supervisors (connectors) and verify that they satisfy the specifications. If the test fails, then the designer has to modify the supervisors till a satisfactory solution is obtained.

**VI. Conclusion**

In this paper the theory of supervisory control of discrete-event systems is used to develop command sequences for control and fault recovery of a spacecraft propulsion subsystem. The study shows that to meet the specifications of normal and faulty modes, the supervisor has to be a “robust” supervisor, and that a conventional (non-robust) supervisor could lead to engine getting stuck in shutdown state. The features of this system that necessitate the use of robust supervisor seem to be present in many aerospace systems. In addition to theoretical developments, we are conducting lab tests on an experimental setup to evaluate the robust supervisory control methodology.

**References**


