Designing a Fault-Tolerant Satellite System in SystemC

Kashif Javed
Department of Information Technologies
Abo Akademi University
Turku, FIN-20520, Finland
Kashif.Javed@abo.fi

Elena Troubitsyna
Department of Information Technologies
Abo Akademi University
Turku, FIN-20520, Finland
Elena.Troubitsyna@abo.fi

***Abstract*—Designing fault-tolerant satellite systems is a challenging engineering task. Often behavior of satellite systems is structured using notion of modes. Ensuring correctness of mode transitions is vital for guaranteeing safe and fault-tolerant functioning of a satellite. In this paper, we propose an approach to designing fault-tolerant satellite systems in SystemC. We demonstrate how to develop Attitude and Orbit Control System in SystemC and verify its correctness via model checking.**

Keywords-component; Fault-Tolerance; Mode-Rich Systems; Design; Verification

I. INTRODUCTION

Designing a system controlling a spacecraft is a challenging engineering task. The system should satisfy a large number of diverse functional and non-functional requirements. In particular, the designers should aim at building a fault-tolerant system, i.e., the system that should cope with faults of various system components. Often behavior of satellite systems is structured using the notion of modes – mutually exclusive sets of system behavior. Fault-tolerance is achieved by putting the system to some downgraded mode when an error occurs. In this paper, we consider an Attitude and Orbit Control System (AOCS) – a generic subsystem of a spacecraft [1]. We demonstrate how to achieve fault-tolerance via backward mode transitions.

AOCS is a complex control system consisting of several components. To ensure correctness of mode transition, we need to guarantee that all components reach a certain state. Moreover, when a component fails we need to guarantee that all other components make an appropriate backward transition.

In this paper, we propose an approach for designing more-rich system in SystemC programming language. We propose an algorithm defining mode-transition scheme of AOCS. To confirm correctness of our algorithm, we have converted it into Promela [6,7] and the results have been verified using SPIN model checker [7,8].

Section II presents architecture of the system. Unit branch state and state transitions have been explained in Section III and the controller phases & phase transitions of the AOCS are described in Section IV. Mode transitions and fault-tolerance procedures for correct functioning of the satellite under faulty conditions are illustrated in Sections V and VI respectively. Section VII explains verification of the implemented system and the paper is summarized in Section VIII besides giving direction for the future work.

II. ARCHITECTURE

The main purpose of AOCS is to control attitude and orbit [1] of a satellite. AOCS consists of a number of components -- AOCS Manager, FDIR (Failure Detection, Isolation and Recovery) Manager, Mode Manager and Unit Manager. The AOCS manager plays key role while dealing with the processing of sensor data, managing actuator movements relating to the units of Reaction Wheel (RW) and Thruster (THR) and doing computation for various controls. The responsibility of FDIR is to timely deal with such tasks as failure detection, isolation and recovery. Mode transitions are handled by the Mode Manager whereas the Unit Manager deals with unit reconfigurations and unit level state transitions [2,3]. Mode and Unit Manager Architectures are further elaborated in the following paragraphs.

A. Mode Manager

The responsibilities of mode include checking of mode transition preconditions, execution of mode transitions, management of controller phases and partially management of related units. There are six different types of controlled modes (i.e. Off, Standby, Safe, Nominal, Preparation and Science) in the mode manager and each mode has its own well-defined unique function. A brief summary of these modes is given below:

1) Off Mode: The satellite is immediately switched in the off mode as soon as the AOCS software booting is completed from the central data management unit.

2) Standby Mode: It is important to check and ensure successful separation of the spacecraft from the launcher and this work is continuously monitored and completed by the software process during the standby mode.

3) Safe Mode: Satellite enters this mode when the separation from the launcher is done. As soon as the system is in the safe mode, the relevant portions of Earth Sensor (ES), RW (Reaction Wheel) and Sun Sensor (SS) are switched to on state, the coarse pointing controller goes in the running phase and fine pointing controller is put in the idle phase. Initially the satellite acquires a stable attitude and then it achieves the coarse pointing.

 ISBN: 978-1-61208-184-7

4) Nominal Mode: When a mode transitions to nominal, the coarse pointing controller becomes idle and the fine pointing controller is set to the running phase. The selected branches of RW, Star Tracker (STR) and THR are switched to on state. In this mode, the satellite utilizes fine pointing control so that the Payload Instrument (PLI) in the AOCS is properly used for measurements.

5) Preparation Mode: The moment the mode is transitioned to the preparation, the concerned portion of Global Positioning System (GPS) is set to fine state, the relevant branch of PLI is switched to standby state and needed processes of RW, STR and THR go to on state. Thus, this mode ensures that the fine pointing control is reached and PLI gets ready for fulfilling its required tasks.

6) Science Mode: In science mode, the selected branch of GPS remains in the fine state, the concerned branch of PLI goes in the science state and the relevant parts of RW, STR and THR maintain their on state. Therefore, the PLI in this mode is ready to perform the tasks for which it has been designed. It stays in this mode till the completion of planned tasks.

B. Unit Manager

The AOCS consists of seven different units and internal state changes in these units are controlled by the unit manager. Mode manager controls the components of unit manager. Seven different controlled units are ES, SS, STR, GPS, RW, THR and PLI. Their brief description is as under:

1) ES is a device that measures the direction to the earth in the sensor's field of view. ES's internal state is either on and off.

2) SS is a tool to measure the direction to the sun in the sensor's field of view. It is also in the on or off state.

3) STR is an optical device that measures the position of stars in its field of view and performs pattern recognition on these stars in order to identify the portion of the sky at which it is looking. Two possible STR's operational states are on and off.

4) GPS is a sophisticated gadget that receives readings related to the satellite position and makes calculations to determine satellite's attitude. Two possible states of GPS operation are coarse navigation and fine navigation.

5) RW is a rotating wheel which is essentially required in order to apply the required torque to the satellite. It is achieved by accelerating or breaking the wheel. RW's state can be either on or off.

6) THR is a position actuator that is used to force the satellite to change its position and its orbit by emitting gas. It can also be in either on or off state.

7) The PLI is an instrument which provides required measurements pertaining to the specific mission. It can operate in standby or science state.

III. UNIT BRANCH STATE AND STATE TRANSITIONS

Every unit is implemented as a pair of identical devices to maintain the nominal branch and the redundant branch. For each unit, one and only one branch is selected at a time. Every selected branch is in on state and its status is locked. In other words, a branch in the off state is always allocated an unlocked status.

In total, there are six states of unit components (i.e. on, off, coarse, fine, standby and science). Whenever an unit state goes from off to on, the powering takes place. Similarly, when the unit switches from on to off state, un-powering takes place. Powering and un-Powering are associated with the states and state transitions of a branch of ES, SS, STR, RW or THR. Occurrence of such states and state transitions is shown in Figure 1. For the GPS unit, unit state goes from off to coarse state and coarse to fine state, then powering and upgrading is carried out respectively. In case of fine to off state transition, first downgrading is performed then un-powering is done. States and State Transitions of a Branch of GPS are depicted in Figure 2.

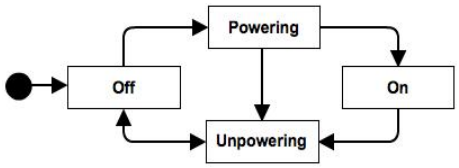

Figure 1: States and State Transitions of a Branch of ES, SS, RW, STR or THR [1]

In case of PLI unit, when the unit state goes from off to standby and from standby to science state, then powering and upgrading is achieved respectively. In case of science to off state transition, first downgrading occurs and then un-powering takes place. Figure 3 demonstrates states and their transitions of a branch of PLI.

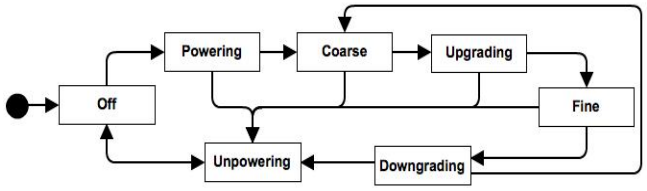

Figure 2:States and State Transitions of a Branch of GPS [1]

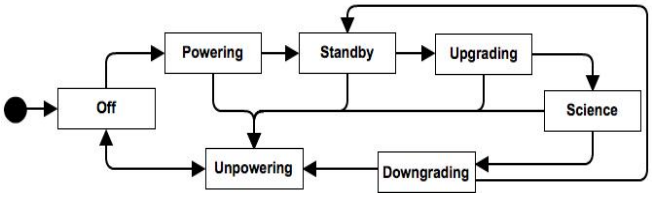

Figure 3: States and State Transitions of a Branch of PLI [1]

State transitions are very fast to accommodate time constrains for real-time satellite operations. Hence, any state transition to powering, un-powering, upgrading or downgrading takes less than one AOCS cycle. However, every state transition to off takes minimum three and maximum four AOCS cycles. Any state transition to on, coarse, fine, standby or science has a success condition if the transition gets completed during the first AOCS cycle when the condition is observed to hold. However, any state transition to on, coarse, fine, standby or science is overridden if the associated success condition is not observed to hold within a predefined number of AOCS cycles from start of the transition.

IV. CONTROLLER PHASES AND PHASE TRANSITIONS

The AOCS has two controllers -- Coarse Pointing Controller (CPC) and Fine Pointing Controller (FPC). The main objective of these two controllers is to direct the line of sight with a specified coarse accuracy and fine accuracy respectively. It is an essential requirement and must be met within given time limits. The following rules have to be observed during the controller phase transitions when a certain operational mode is reached:

1) Both controllers go to idle phase when the mode transition is set to off or standby state.

2) When the mode transition is switched to safe state, the CPC enters the running phase and the FPC remains in the idle phase.

3) When the mode transition shifts to nominal, preparation or science, the CPC goes in the idle phase and the FPC moves in the running phase.

Only one controller can be in non-idle phase at any point of time. When a controller phase has to switch from idle to running, first of all it is set to preparing. After predefined number of AOCS cycles, the controller is set to ready phase. Finally, the phase of controller is shifted to running as indicated in Figure 4. It can also be noticed that the controller can directly move to the idle phase from any of the other three phases (preparing, ready and running).

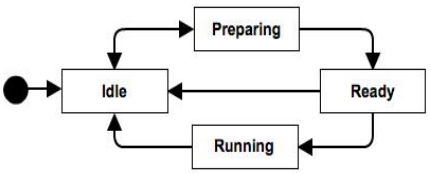

Figure 4: Phases and Phase Transitions of a Controller [1]

V. MODE TRANSITIONS

The following rules are imposed on mode transitions in order to ensure correct satellite function in nominal (fault-free) and faulty conditions:

1) When a mode transition to off or standby is completed, it is ensured that every branch in every unit is put in the off state.

2) On reaching to the safe mode, the selected branches of ES, RW and SS are set in the on state and all other branches pertaining to different units go to the off state.

3) In case of a transition to the nominal mode, the selected branch of GPS is turned in the coarse state, the concerned branches of RW, STR and THR are set to on state, and remaining every branch in every unit is put in the off state.

4) Completion of a mode transition to preparation ensures that the relevant branch of GPS is in the fine state, the chosen branch of PLI is in the standby state, the selected parts of RW, STR and THR are in the on state, and rest every branch in every unit is in the off state.

5) A mode transition to science requires that the needed branch of GPS is in the fine state, the selected branch of PLI is in the science state, the concerned branches of RW, STR and THR are in the on state, and all other branches pertaining to different units remain in the off state.

VI. FAULT TOLERANCE

Fault-tolerance should guarantee that the system continues to operate in predictable way even in case of failure of any of its components. Recovery from errors in fault-tolerant systems can be characterized as either roll forward or roll back. Forward error recovery aims at bringing the system to a new error-free state. Backward error recovery rolls back the system to some previous state before an error occurrence. In mode-rich systems, the backward error recovery is achieved via backward mode transition, i.e., mode downgrading. The mode down-gradation depends on various errors, which are explained below:

A. Branch State Transition Errors

A branch state transition error means that when some unit transitions to on state, the mode coarse, fine, standby or science gets overridden due to timeout condition. Because operation and state transition delays have to be avoided, we should time each mode transition. If a step of transition is not completed within a specified time limit, timeout signal is generated to get into a safe condition. The important error checks concerning to the branch state transitions are:

1) A branch state transition error on the redundant branch of ES, RW or SS causes a mode transition to off.

2) A mode transition to safe takes place when there is a branch state transition error on the redundant branch of GPS, STR or THR and there is no branch state transition error on the redundant branches of ES, RW and SS.

3) When a branch state transition error on the redundant branch of PLI occurs, it results into a mode transition to

 ISBN: 978-1-61208-184-7

nominal provided that there is no branch state transition error on the redundant branches of ES, SS, GPS, RW, STR and THR.

B. Phase Transition Errors

A phase transition error or an attitude error may arise during the computations done by the selected controller. An attitude error is generated when there is a problem in the execution of an AOCS algorithm. It means that an error occurs only when one of the two controllers (i.e. CPC and FPC) is in the running phase. The key factors relating to the attitude errors are:

1) If the current mode is safe, then a non-ignored attitude error causes a transition to the off mode.

2) In case the existing mode is nominal and a non-ignored attitude error occurs, a mode transition to safe takes place.

3) A mode transition to nominal takes place when the current mode is preparation and a non-ignored attitude error is generated.

4) The generation of a non-ignored attitude error moves the mode transition to preparation with the condition that the existing mode is science.

C. Unit Reconfiguration

Each logical unit consists of two hardware units known as nominal and redundant. Initially, the nominal unit works in the active role and provides all the necessary support for normal operation of the system. The redundant unit serves as a backup resource. When an error is detected in the nominal unit, it becomes “reconfigured”. It means that the nominal unit is switched off and the redundant unit takes over the operational tasks.

The important errors that take place during the unit reconfiguration are:

1) A branch state transition error on the nominal branch of ES, SS or RW causes a reconfiguration of the unit if there is no branch state transition error on the redundant branches of ES, SS and RW.

2) A branch state transition error on the nominal branch of GPS, STR, THR or PLI causes a reconfiguration of the unit if there is no branch state transition error on the redundant branches of ES, SS, GPS, RW, STR and THR.

Figure 6 shows detailed flow chart of the implemented system.

VII. VERIFICATION

We have implemented mode-transition algorithm in SystemC language. The SystemC Verification Standard provides API for transaction based verification, constrained and weighted randomization, exception handling, and other verification tasks [4,5]. SystemC supports the use of special data types which are often used by the hardware engineers. It comes with a strong simulation kernel to enable the designers to write good test benches for easy and speedy simulation. It is extremely important because the functional verification at the system level saves a lot of money and time.

The system architecture that is implemented in SystemC is verified in the SPIN model checker. SPIN [6,7,8] is often used to verify behavior of distributed and parallel systems. PROMELA (PROcess MEta LAnguage) is a high level language which is widely used to specify systems descriptions and is fully supported by SPIN for the purpose of verification of software-based applications. SPIN PROMELA is used to carry out detailed testing and verification of design and architecture of various systems.

The simplified system architecture for AOCS is shown in Figure 5.

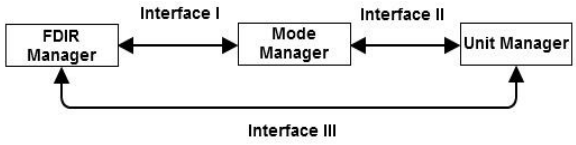

Figure 5: System Architecture [1]

An example of an interfaces between the FDIR Manager, Mode Manager and Unit Manager shown in Figure 5 are given below.

When failure occurs in the system, FDIR detects the error and issues the requests of mode transition, and then Mode Manager is responsible for mode transitions to the downgraded mode on the basis of error type. The following part of the code represents the Interface I scenario for Science Mode.

```
if (Mode==F) // Mode F: Science Mode
{       if (ES==off && SS==off && GPS==fine && STR==on &&
        RW==on && THR==on && PLI==science && CPC==idle
        && FPC==run)
        {/* The associated code describes that the conditions are valid
        for Science Mode. The current mode is Science. */}
        else if ((ES!=off || SS!=off || RW!=on) && STR==on &&
        GPS==fine && THR==on && PLI==science && CPC==idle
        && FPC==run)
        {/* The associated code describes that the conditions are not
        valid for Science Mode as error occurs on the unit branch of ES,
        SS or RW. It causes the mode transition to Off Mode. */}
        else if ((GPS!=fine || STR!=on || THR!=on) && ES==off &&
        SS==off && RW==on && PLI==science && CPC==idle &&
        FPC==run)
        {/* The associated code describes that the conditions are not
        valid for Science Mode as error occurs on the unit branch of
        GPS, STR or THR. It causes the mode transition to Safe Mode.
        */}
        else if (ES==off && SS==off && GPS==fine && STR==on
        && RW==on && THR==on && PLI!=science && CPC==idle
        && FPC==run)
        {/* The associated code describes that the conditions are not
        valid for Science Mode as error occurs on the unit branch of
        PLI. It causes the mode transition to Nominal Mode. */}
        else if (ES==off && SS==off && GPS==fine && STR==on
        && RW==on && THR==on && PLI==science &&
        (CPC!=idle || FPC!=run))
        {/* The associated code describes that the conditions are not
        valid for Science Mode as error occurs in the phase of Coarse or
```

```
        Fine Pointing Controller. It causes the mode transition to
        Preparation Mode. */}
        else
        {/* The associated code describes that no transitions take place.
        */ } }
    else
    {/* The associated code describes that it is an invalid mode. Program is
terminated.*/}
```

The SPIN's verification model successfully checks all the global mode transitions and the fault-tolerance of the system architecture. We have successfully verified forward and backward mode transitions and ensured correctness of global mode transitions with respect to component states.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed an approach to designing fault tolerant mode-rich control systems. Our work aimed at demonstrating how to design satellite control system in SystemC and verify correctness using model checking. Our approach has been demonstrated by the design of Attitude and Orbit Control System – a generic subsystem of spacecrafts.

The proposed system has been implemented in SystemC language as it is being used as a defacto verification standard in embedded systems. SystemC specification was easily aligned with Promela which works as the input language to SPIN for model checking and verification.

We have presented the design of the system and verification steps pertaining to unit branch transition errors, controller phase transition errors and unit reconfiguration.

Our work complements research done on formal modeling of mode-rich satellite systems. The formal modeling undertaken in [9,10] aimed at enabling proof-based verification of mode-rich systems modeled in Event-B. In [11] the authors perform failure modes and effect analysis of each particular mode transition to systematically design mode transition scheme. Our work aims at building a gap between formal specification and code. This motivated our choice of SystemC as a design language and model-checking based verification.

As a future work, we are planning to investigate design and verification of decentralized mode-rich systems. In particular, we will study how to ensure correctness of mode transitions as a result of negotiation between several mode managers.

REFERENCES

[1] "DEPLOY Work Package 3 - Attitude and Orbit Control System Software Requirements Document", Space Systems Finland, Ltd., December 2010.

[2] M. Heimdahl and N. Leveson, "Completeness and Consistency in Hierarchical State-Based Requirements", IEEE Transactions on Software Engineering, Vol.22, No. 6, June 1996, pp. 363-377.

[3] N. Leveson, L. D. Pinnel, S. D. Sandys, S. Koga, and J. D. Reese, "Analyzing Software Specifications for Mode Confusion Potential", Proceedings of Workshop on Human Error and System Development, C.W. Johnson, Editor, March 1997, Glasgow, Scotland, pp. 132-146.

[4] C. Ip and S. Swan, "A tutorial introduction on the new SystemC verification standard", Technical report, www.systemc.org, 2003.

[5] L. Singh and L. Drucker, "Advanced Verification Techniques : A SystemC Based Approach for Successful Tapeout", Springer, 2004.

[6] J. Katoen, "Concepts, Algorithms and Tools for Model Checking", Lecture Notes, Chapter 1: System Validation, 1999.

[7] N. A. S. A. Larc, "What is Formal Methods?", NASA Langley Methods, http://shemesh.larc.nasa.gov/fm/fm-what.html, formal methods program, 2001.

[8] Kashif Javed, Asifa Kashif, and Elena Troubitsyna, "Implementation of SPIN Model Checker for Formal Verification of Distance Vector Routing Protocol", International Journal of Computer Science and Information Security (IJCSIS), Vol 8, No 3, June 2010, USA, ISSN 1947-5500, pp. 1-6.

[9] Alexei Iliasov, Elena Troubitsyna, Linas Laibinis, Alexander Romanovsky, Kimmo Varpaaniemi, Dubravka Ilic, and Timo Latvala. Developing Mode-Rich Satellite Software by Refinement in Event B . In Proceedings of FMICS 2010, the 15th International Workshop on Formal Methods for Industrial Critical Systems, September 2010, LNCS 6371. Springer.

[10] Alexei Iliasov, Elena Troubitsyna, Linas Laibinis, Alexander Romanovsky, and Kimmo Varpaaniemi, Pauli Väisänen. Verifying Mode Consistency for On-Board Satellite Software, 2010, LNCS 6351, Computer Safety, Reliability, and Security, Pages 126-141, Springer.

[11] Yuliya Prokhorova, Elena Troubitsyna, Linas Laibinis, Kimmo Varpaaniemi, and Timo Latvala. Derivation and Formal Verification of a Mode Logic for Layered Control Systems. Asia-Pacific Software Engineering Conference. IEEE Computer, December 2011.

 ISBN: 978-1-61208-184-7

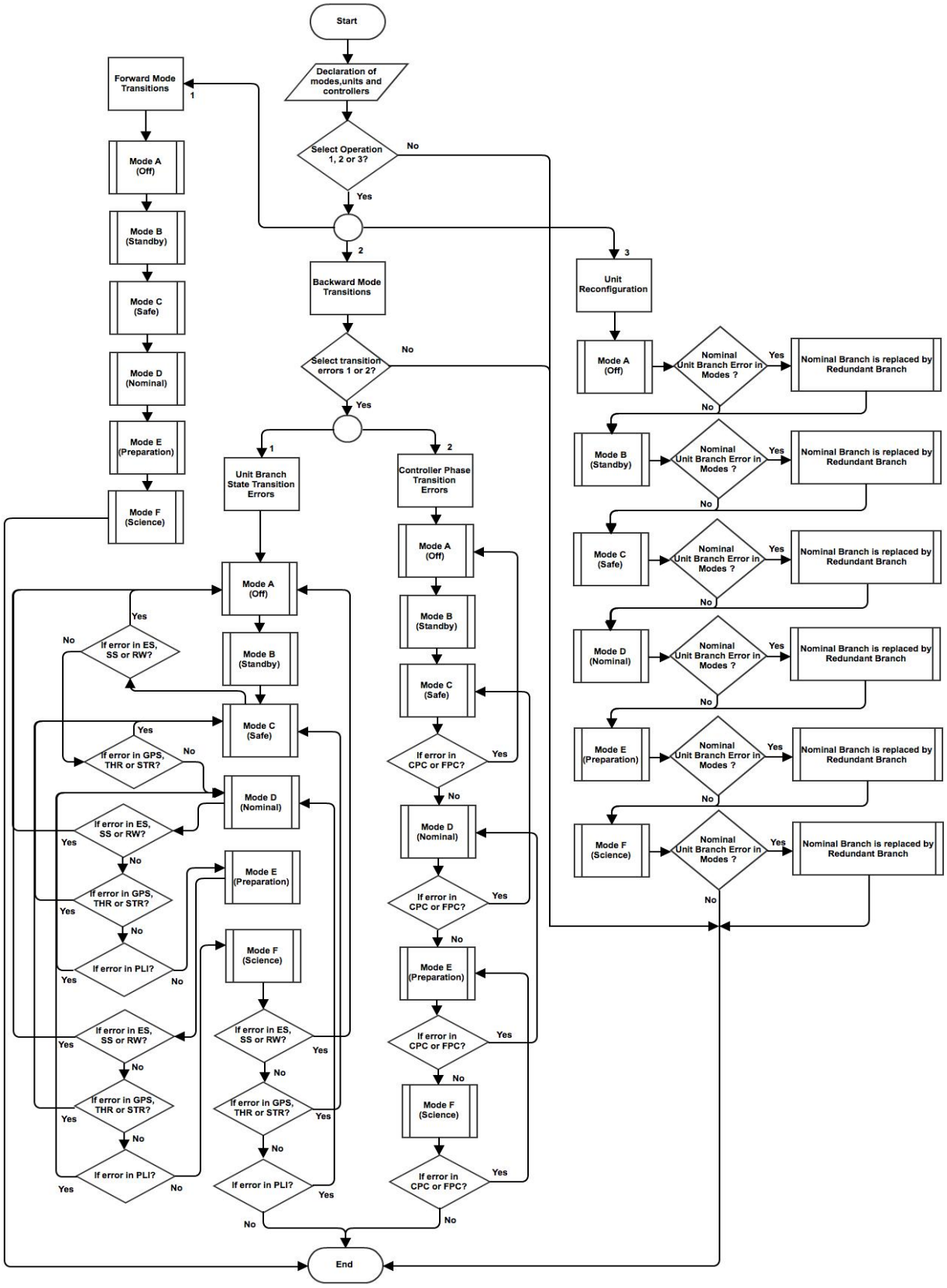

Figure 6: System Flow Chart