West Virginia University
College of Engineering and Mineral Resources

Lane Department of Computer Science and Electrical Engineering

Real Time Systems
Spring 2008

Automated Commuter Train System
Analysis Phase

John Sedoski
Tobias Brozenick
Anthony Graziani

Table of Contents

Table of Figures:

1: Scope:

1.1: Identification and System Overview:

The scope of the Automated Commuter Train System (ACTS) is defined by a number of requirements and features. The ACTS is fully automated, while still allowing a train operator to assume control when necessary. The responsibilities of the Automated Commuter Train System include:

- Monitoring external component sensors (doors, temperature, brakes, engine, etc).
- Watching for external component failures.
- Scheduling the train start and stop locations based on a specific scheduling mode.
- Running the automated system in accordance with the sensor and scheduling data.

Other features available in the ACTS are PA system support, climate control, and light control.

1.2: Document Overview:

The following document is comprised of multiple diagrams and tables that analyze the ACTS. Under the "Required States and Modes" section is a state-transition diagram that illustrates the *necessary* states and modes of the system. A brief description of each state/mode is also included.

The "CSCI Capability Requirements" section contains the Data-Flow Diagrams (DFD) of the system with the appropriate Control Specifications (C-Specs).

The "CSCI External Interface Requirements" section includes the Context Diagram and a brief explanation of the external entities that the ACTS interacts with.

The "CSCI Interface Requirements" section is comprised of the Process Specifications (P-Specs) of all of the primitive functions of the ACTS.

The "CSCI Internal Data Requirements" section includes how the information in the data stores within the ACTS is kept.

The “Data Dictionary Entries” section contains all of the control and data inputs in the system and how they are arranged and passed through the various processes and control nodes.

3: Requirements:

3.1: Required States and Modes:

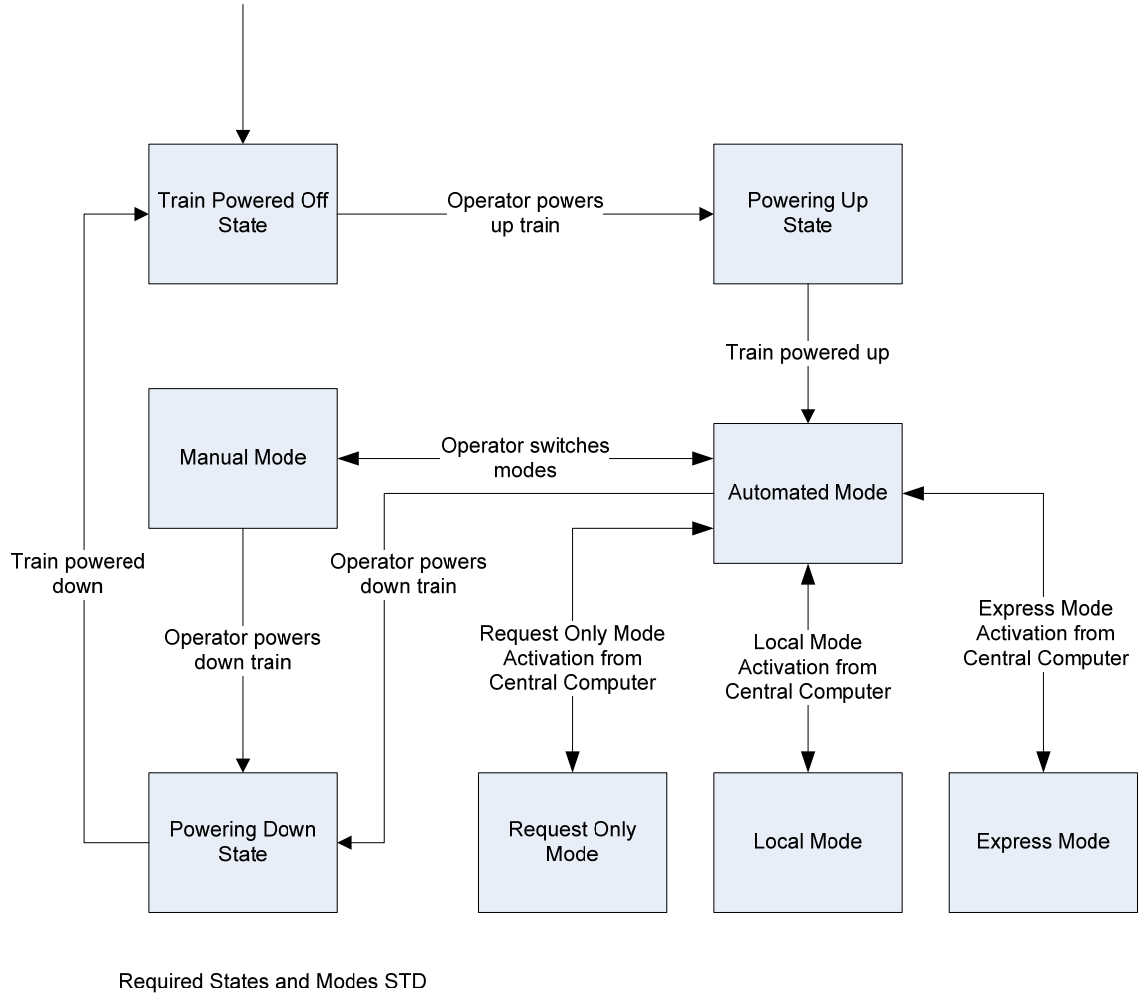


Figure 1 - Required States and Modes STD

Figure 1 depicts the required states and modes for the Automated Commuter Train System. The required states and modes for the Automated Commuter Train begin with the *Train Powered Off State*. When the operator presses the power-up switch, the system goes into the *Powering Up State*. Once the train has successfully powered-up, the system goes right into the *Automated Mode*. The system then has three choices: *Request Only Mode*, *Local Mode*, and *Express Mode*. The system may jump between these three modes at any time as the Central Computer

determines the correct mode. The Train Operator may also override the system by switching from the *Automated Mode* to the *Manual Mode* at any time. Finally, the Train Operator can press the power-down switch to take the system into the *Powering Down State*. After the train has powered-down, the system returns to the original *Train Powered Off State*.

3.2: CSCI Capability Requirements – DFD 0

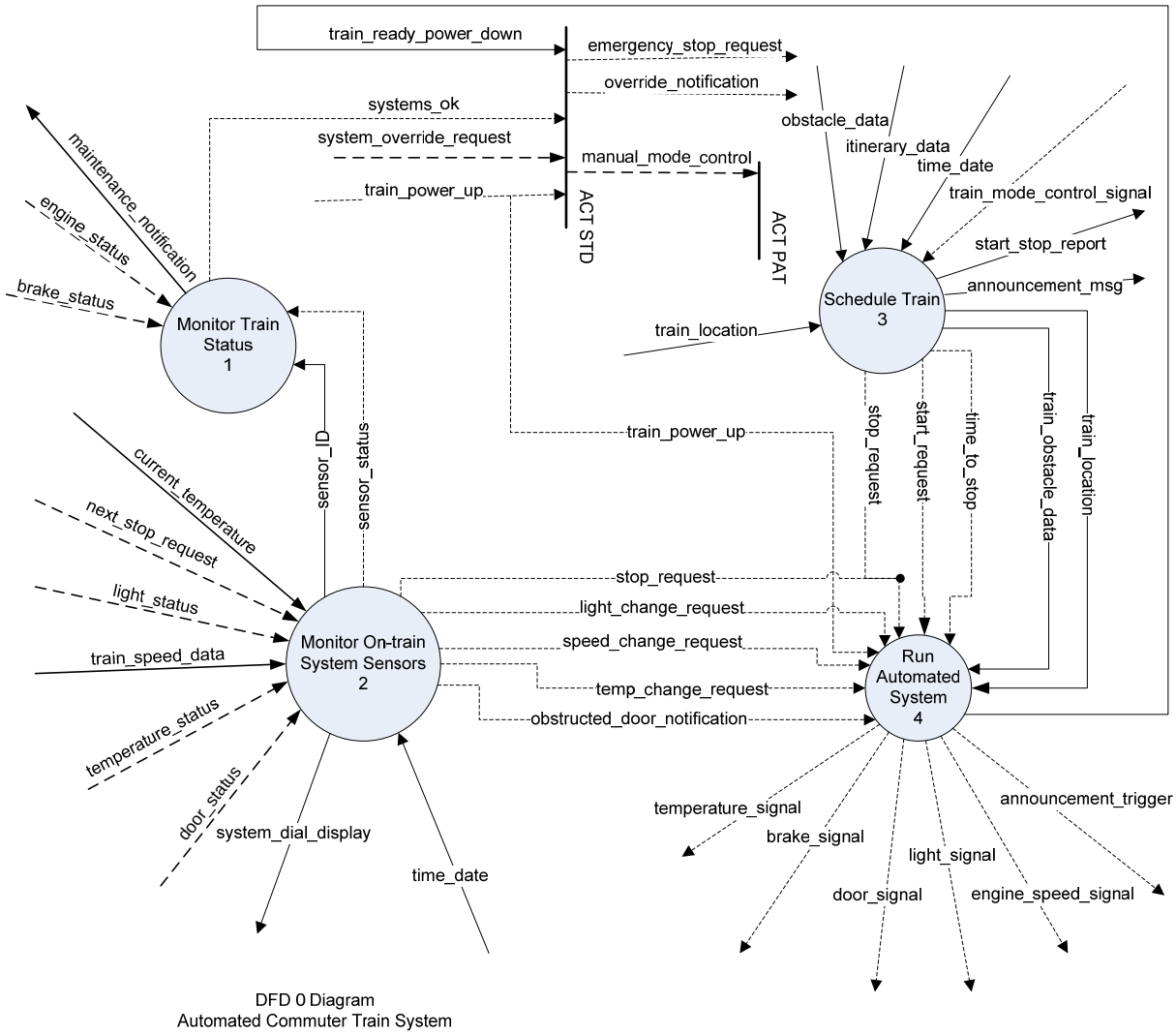


Figure 2 - DFD 0 Diagram: Automated Commuter Train System

The DFD 0 Diagram in Figure 2 is illustrated by the four major functions of the Automated Commuter Train System described below:

1. *Monitor Train Status* – This function is used to monitor the external components for failures. If the brakes, engine, lights, heating system, or doors are not working properly, the Monitor Train Status process will notify the Maintenance System of the failure. An external component failure will also generate a control output, systems_ok, to be described by the ACT STD.

2. *Monitor On-train System Sensors* – This function is used to read the various external sensors and generate an operator dial of the sensor readings. If the system is operating in the Automated Control Mode, then this process will make automated requests to the train system based on the sensor readings.

3. *Schedule Train* – This function is used to schedule the starts and stops of the train based on the train_control_mode input. The itinerary_data, time_date, train_location, and obstacle_data inputs are assessed to determine when and where the train should stop as it follows the proper route. The announcement_msg output is sent to notify passengers of these departures and arrivals via the external PA system. Also, a start_stop_report output is sent to the Maintenance System Database at the end of each train run to account for all of the starts and stops of the run. This function is only activated under the Automated Control Mode.

4. *Run Automated System* – This function is used to automatically process all of the On-train System requests. The automated train requests managed include: stopping and starting the train, opening and closing the doors, maintaining the proper train speed, changing the train lights, and turning on the heater and air conditioner. The Run Automated System process interacts heavily with the Schedule Train and Monitor On-train System Sensors processes. Run Automated System is only activated under the Automated Control Mode.

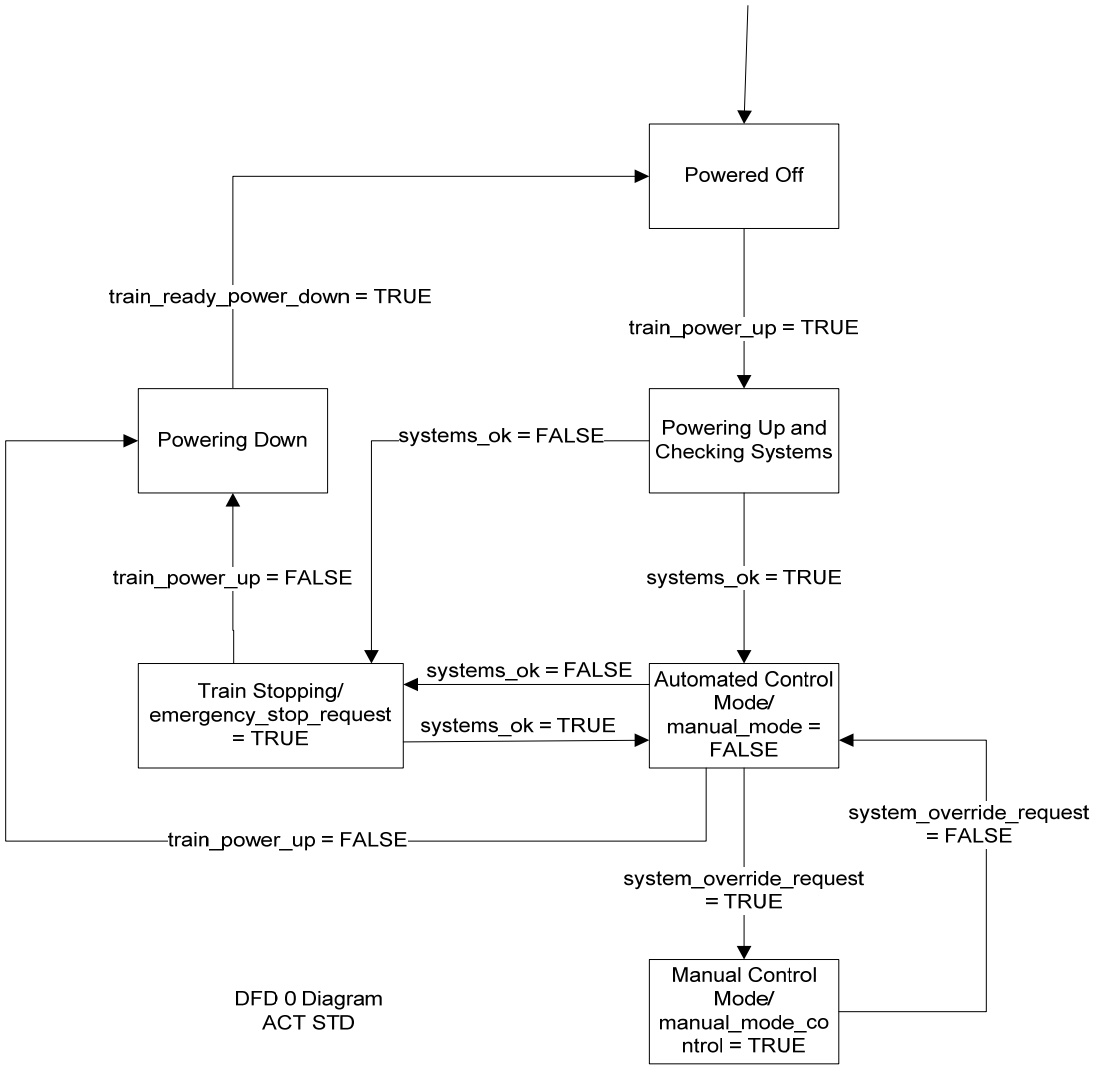


Figure 3 - DFD 0 Diagram: ACT STD

Figure 3 depicts the state-transition diagram for the ACT STD control node. The diagram illustrates how the system goes from the Automated Control Mode to the Manual Control Mode based on the inputs and outputs of the control node.

ACT PAT	Data Processes			
manual_mode_control	**1**	**2**	**3**	**4**
FALSE	ON	ON	ON	ON
TRUE	ON	ON	OFF	OFF

Figure 4 - DFD 0 Table: ACT PAT

Figure 4 shows which data processes are activated when manual_mode_control is changed by using a Process Activation Table (PAT). Note that processes 3 and 4 are deactivated when manual_mode_control = TRUE.

3.2.1: CSCI Capability Requirements – DFD 1

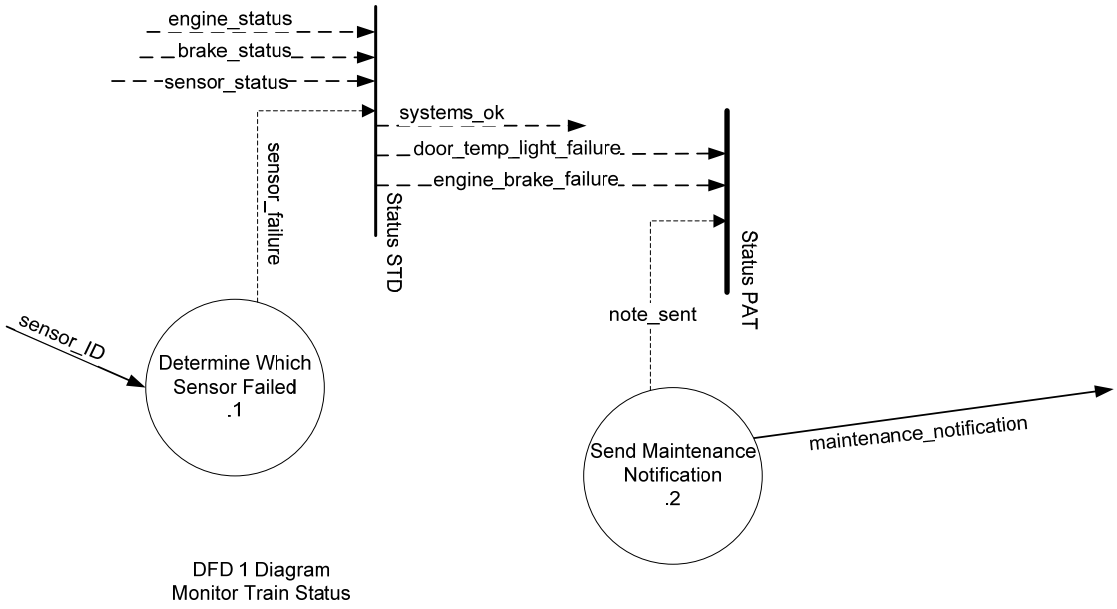


Figure 5 - DFD 1 Diagram: Monitor Train Status

The DFD 1 Diagram in Figure 5 is illustrated by two primitive functions and two control nodes. The control nodes are briefly described below.

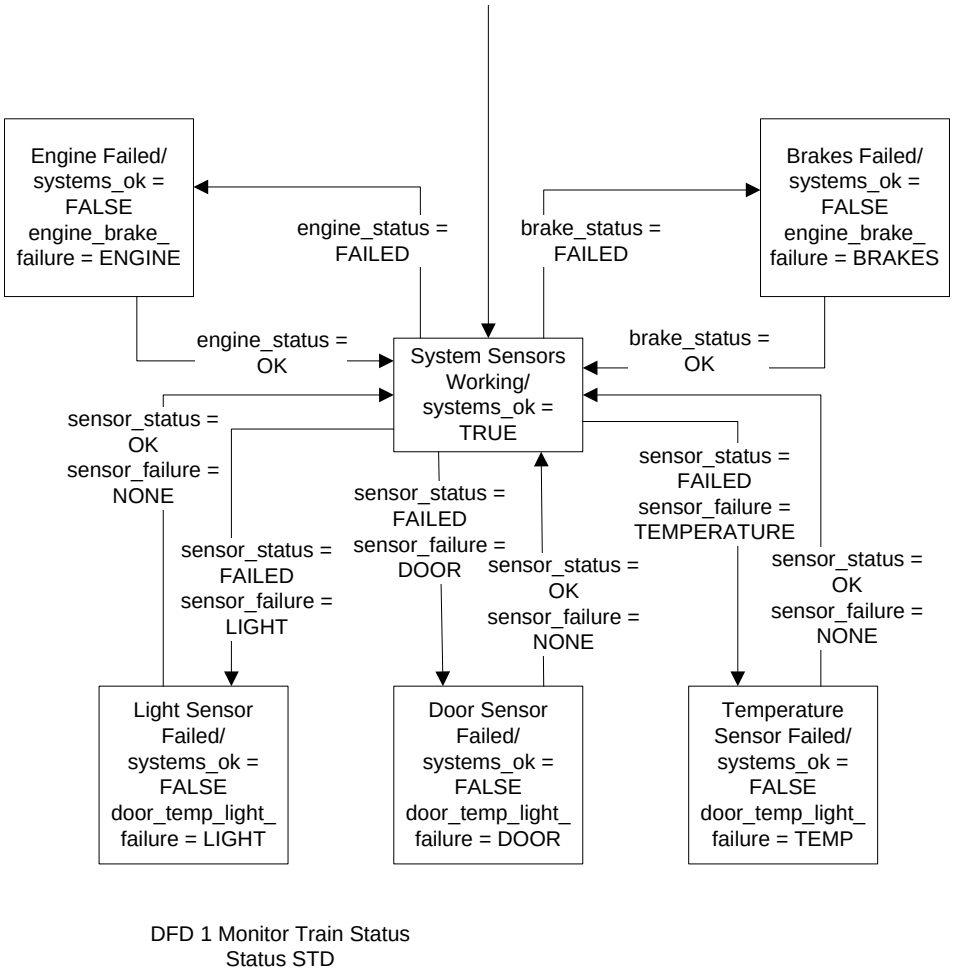


Figure 6 - DFD 1 Diagram: Status STD

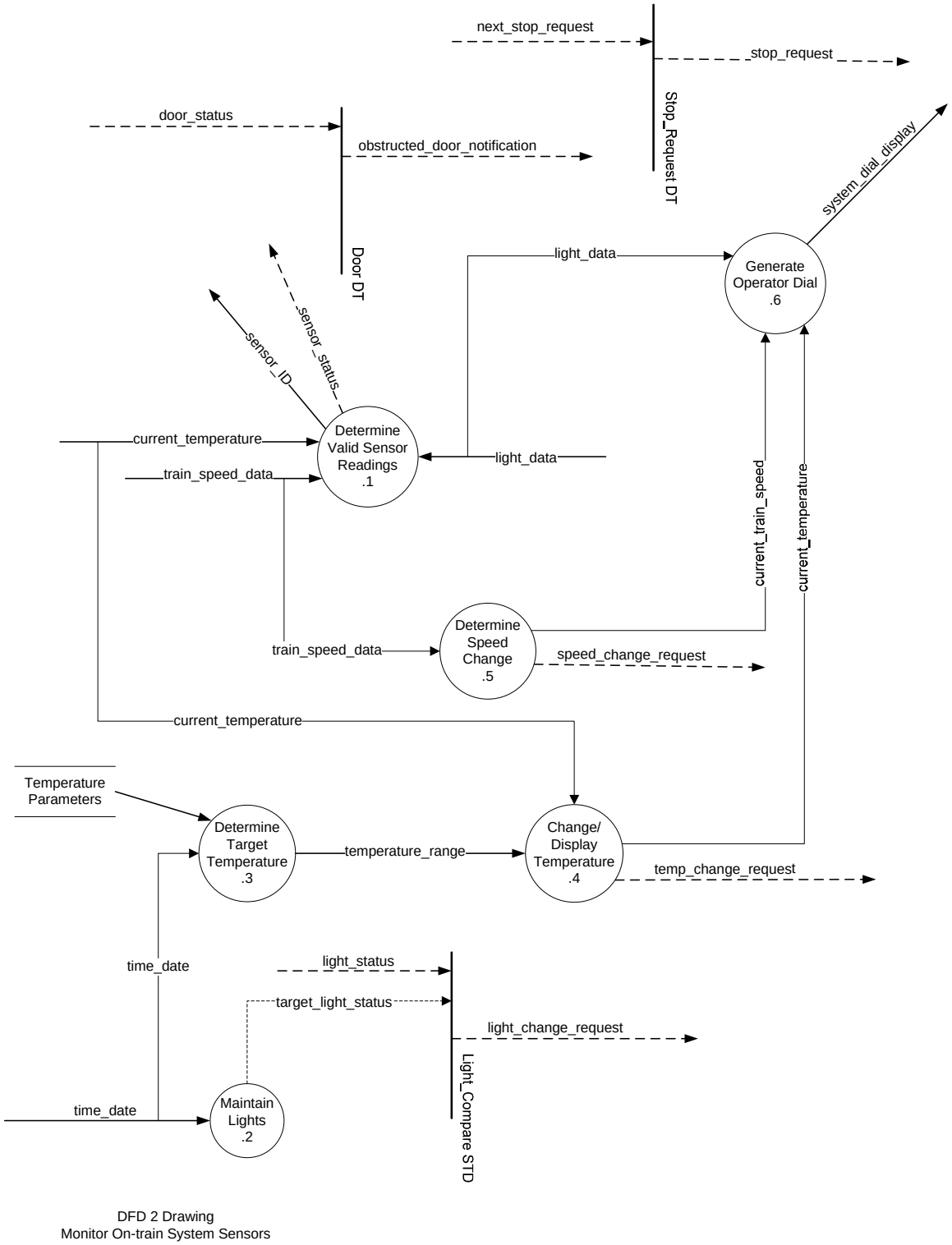
Figure 6 depicts the state-transition diagram for the Status STD control node. The diagram illustrates how the system reacts when an external sensor fails.

DFD 1 Monitor Train Status - Status PAT			Data Processes	
door_temp_light_failure	engine_brake_failure	note_sent	1.1	1.2
DOOR	X	X	X	ON
TEMP	X	X	X	ON
LIGHT	X	X	X	ON
X	ENGINE	X	X	ON
X	BRAKE	X	X	ON
NONE	NONE	TRUE	ON	OFF

Figure 7 - DFD 1 Table: Status PAT

Figure 7 uses a PAT to show which data processes are activated in the event of a door, temperature, light, engine, or brake failure. The input, note_sent, is used to let the system know that the necessary notification has been sent to the Maintenance System.

3.2.2: CSCI Capability Requirements – DFD 2



DFD 2 Drawing
Monitor On-train System Sensors

Figure 8 - DFD 2 Diagram: Monitor On-train System Sensors

The DFD 2 Diagram in Figure 8 is illustrated by six primitive functions and three control nodes. The control nodes for Monitor On-train System Sensors are briefly described below.

DOOR DT - Monitor On-train System Sensors

door_status	obstructed_door_notification
OBSTRUCTED	TRUE
CLEAR	FALSE

Figure 9 - DFD 2 Table: Door DT

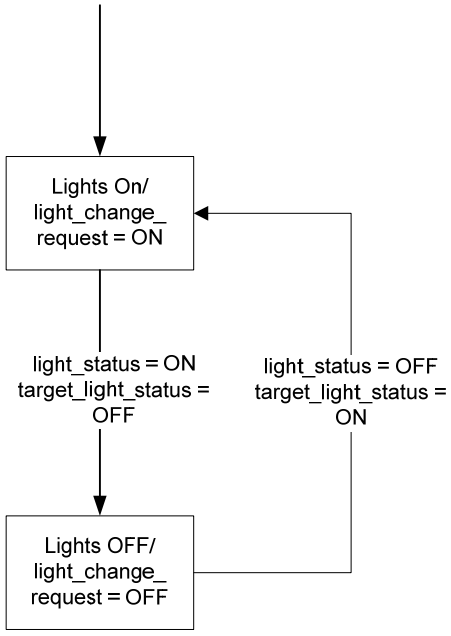
Figure 9 uses a Decision Table (DT) to show the control output in the event of an obstructed door and a clear door. The input, door_status, is used to let the system know whether the door being monitored is obstructed or not.

Stop_Request DT - Monitor On-train System Sensors

next_stop_request	stop_request
PUSHED	TRUE
NOT_PUSHED	FALSE

Figure 10 - DFD 2 Table: Stop_Request DT

Figure 10 uses a DT to show the control output in the event of a passenger pushing the “Next Stop” button. The input, next_stop_request, is used to let the system know whether a passenger wishes the train to stop at the next stop location.



DFD 2 Monitor On-train System Sensors
Light_Compare STD

Figure 11 - DFD 2 Diagram: Light_Compare STD

Figure 11 depicts the state-transition diagram for the Light_Compare STD control node. The diagram illustrates how the system reacts when the light_status is not equal to the target_light_status.

3.2.3: CSCI Capability Requirements – DFD 3

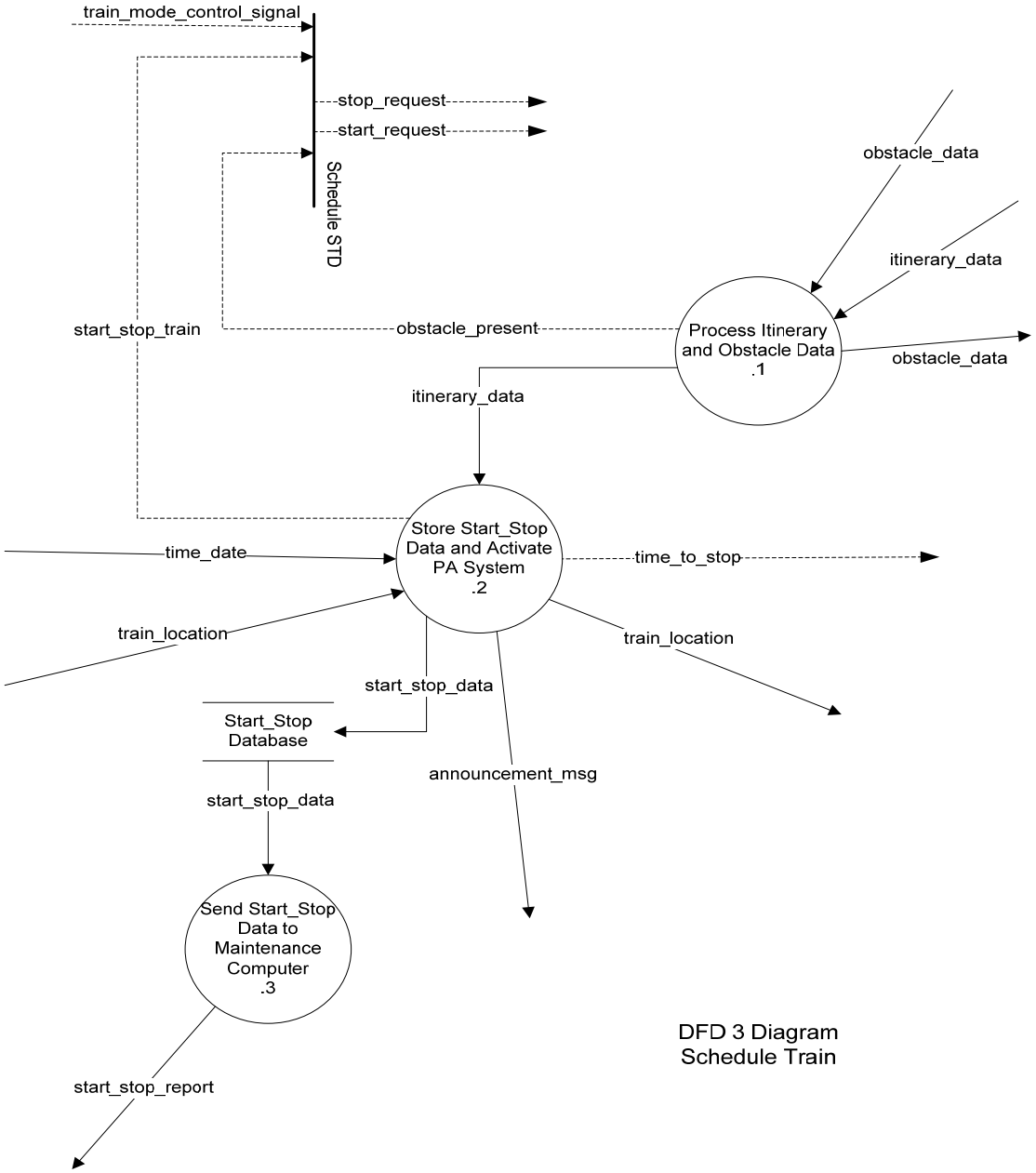


Figure 12 - DFD 3 Diagram: Schedule Train

The DFD 3 Diagram in Figure 12 is illustrated by three primitive functions and one control node. The control node for Schedule Train is briefly described below.

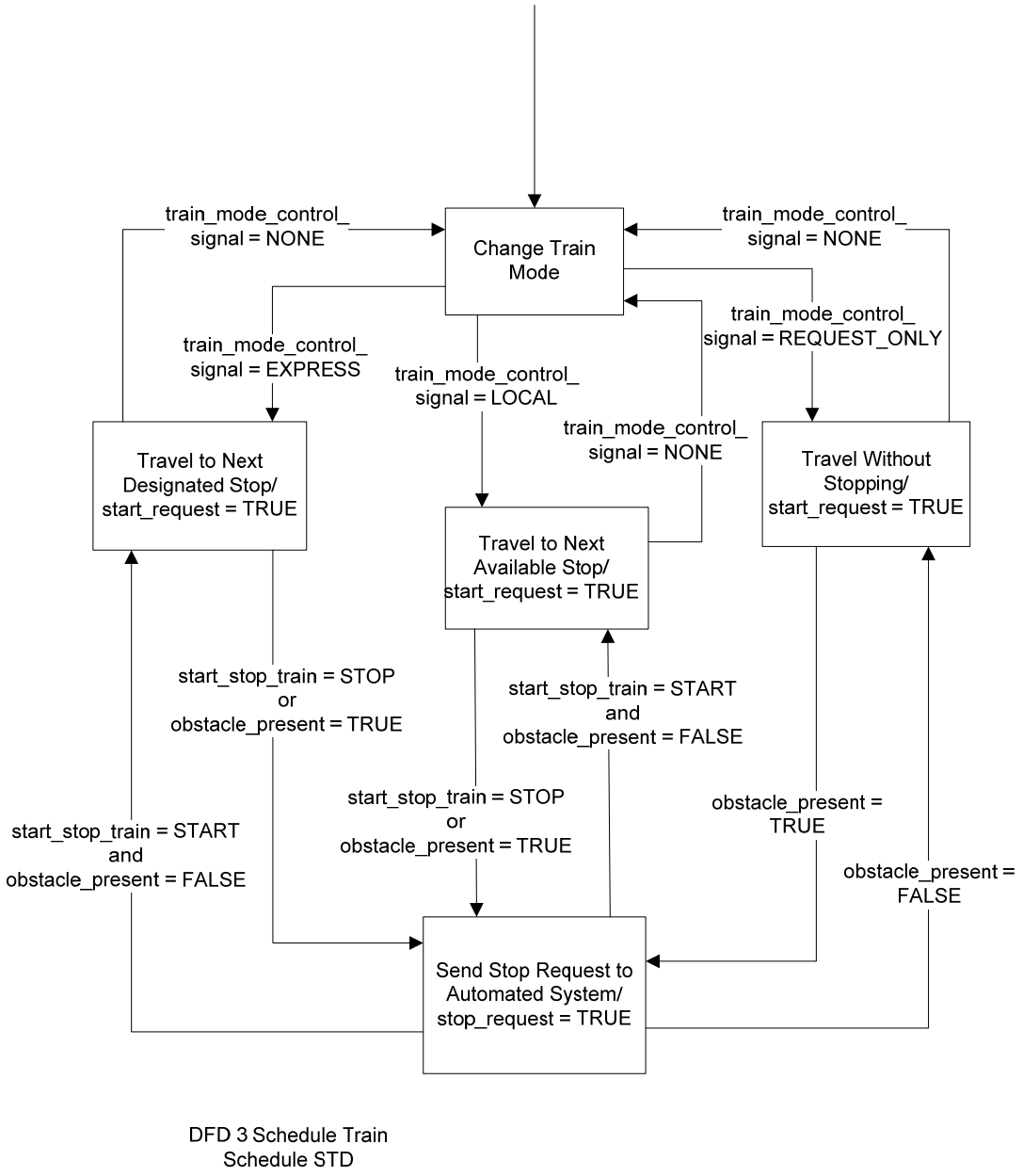


Figure 13 - DFD 3 Diagram: Schedule STD

Figure 13 depicts the state-transition diagram for the Schedule STD control node. The diagram illustrates how the system reacts as the control modes are changed from EXPRESS, LOCAL, and REQUEST_ONLY by the Central Computer. Also, start and stop requests and track obstacles are accounted for during transit.

3.2.4: CSCI Capability Requirements – DFD 4

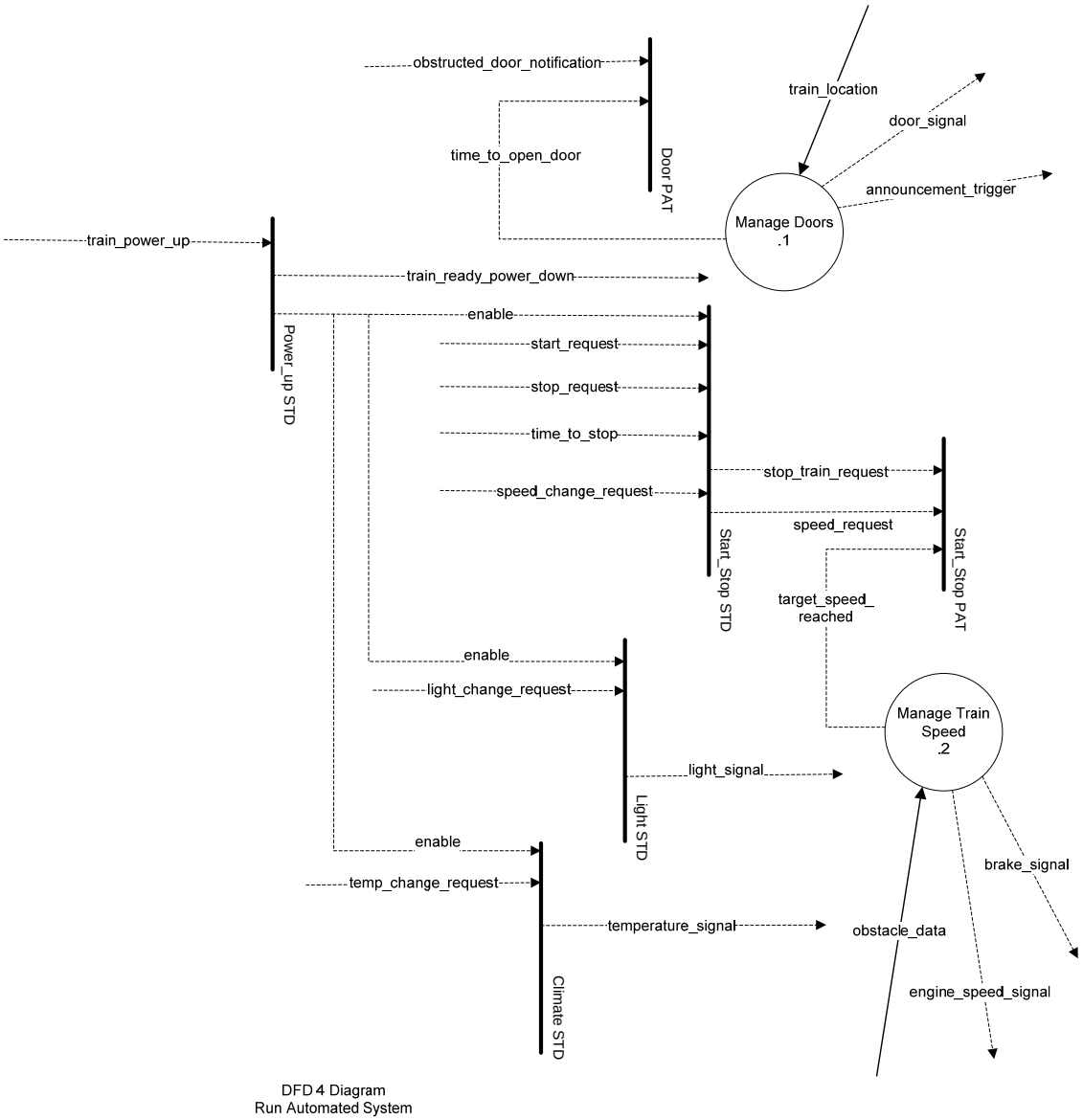


Figure 14 - DFD 4 Diagram - Run Automated System

The DFD 4 Diagram in Figure 14 is illustrated by two primitive functions and six control nodes. The control nodes for Run Automated System are briefly described below.

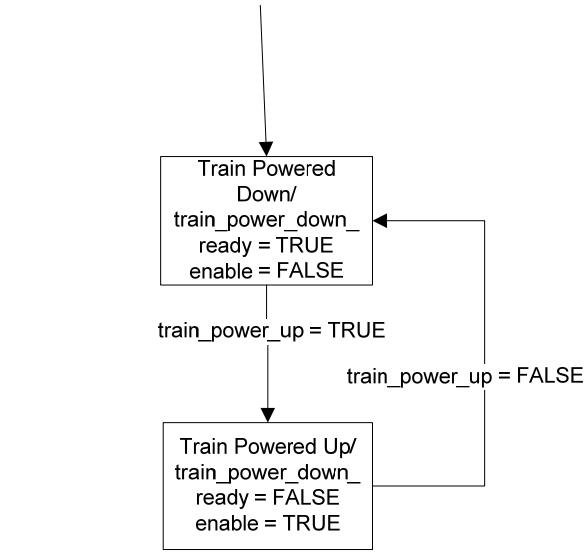


Figure 15 - DFD 4 Diagram: Power_up STD

Figure 15 depicts the state-transition diagram for the Power_up STD control node. The diagram illustrates how the system is powered-up and powered-down by the Train Operator.

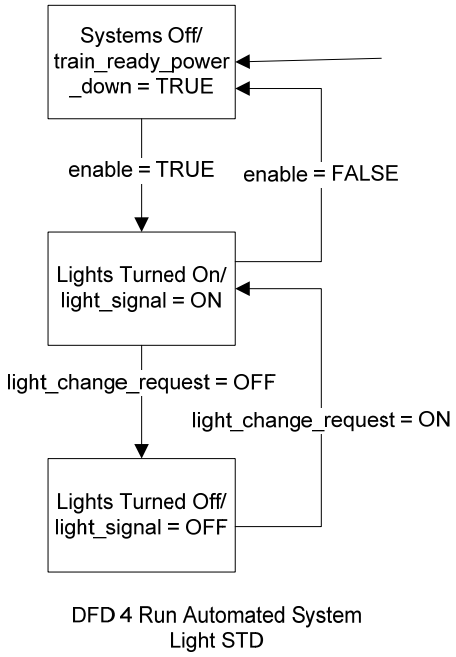


Figure 16 - DFD 4 Diagram: Light STD

Figure 16 depicts the state-transition diagram for the Light STD control node. The diagram illustrates how the system handles light-change requests.

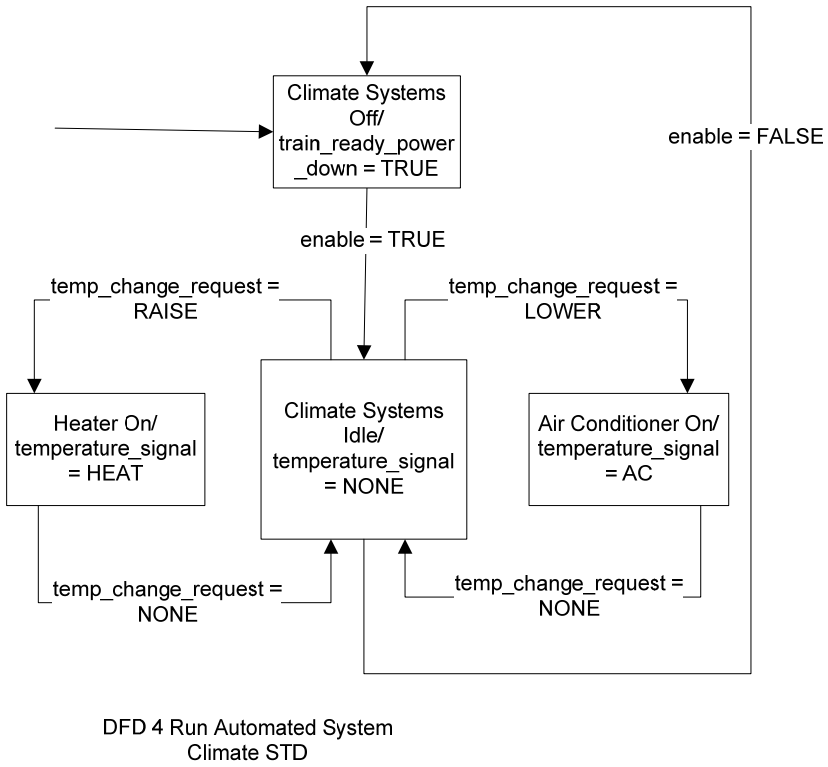


Figure 17 - DFD 4 Diagram: Climate STD

Figure 17 depicts the state-transition diagram for the Climate STD control node. The diagram illustrates how the heater and air conditioner is turned on when temperature-change requests are made by the automated system.

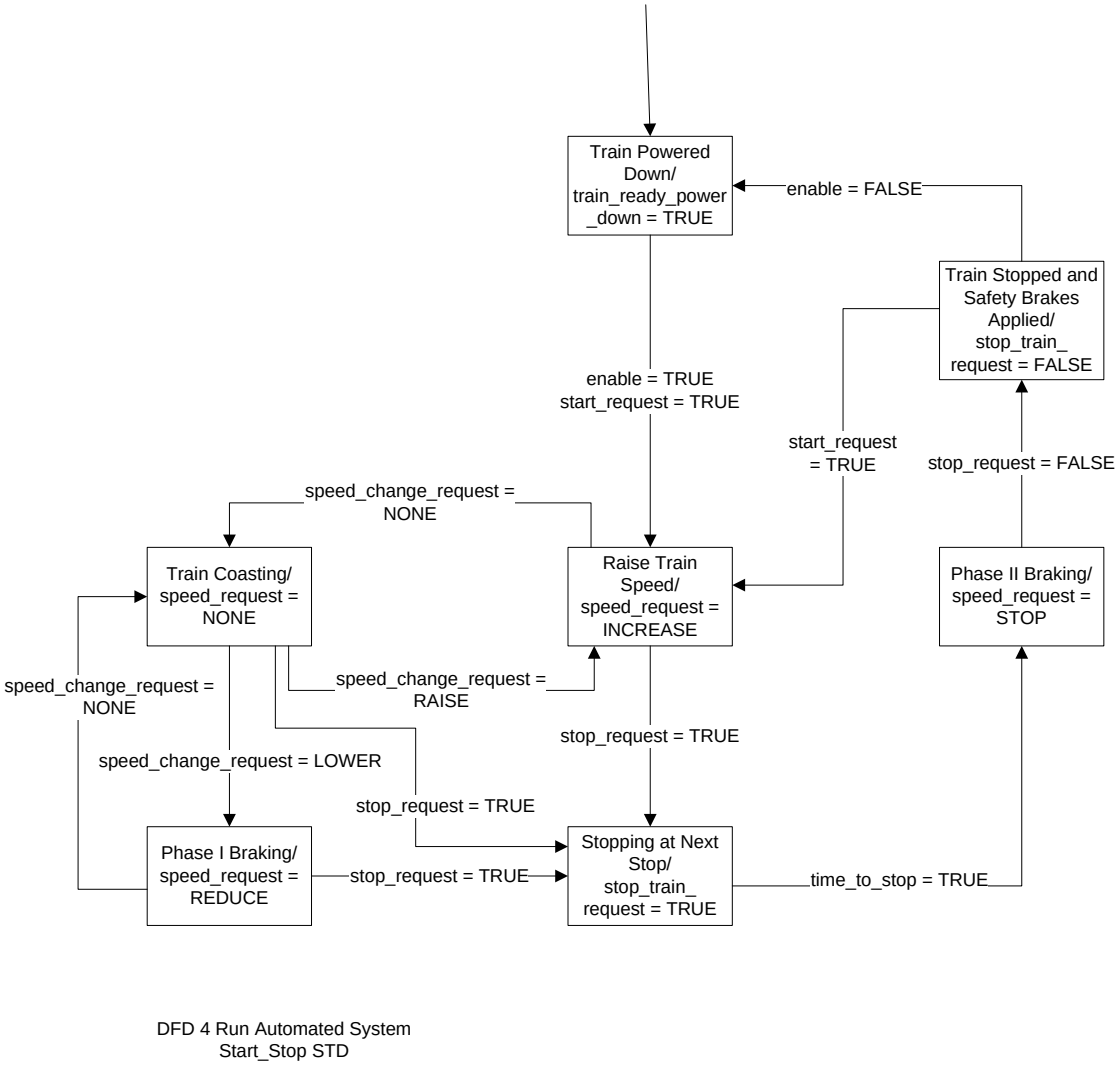


Figure 18 - DFD 4 Diagram: Start_Stop STD

Figure 18 depicts the state-transition diagram for the Start_Stop STD control node. The diagram shows how the system reacts to speed-change requests. Also, the way the system interfaces with the Phase 1, Phase 2, and Emergency brakes is illustrated.

DFD 4 Run Automated System - Door PAT **Data Processes**

obstructed_door_notification	time_to_open_door	4.1	4.2
TRUE	X	ON	X
X	TRUE	ON	X
FALSE	FALSE	OFF	X

Figure 19 - DFD 4 Table: Door PAT

Figure 19 uses a PAT to show which data processes are activated when obstructed_door_notification and time_to_open_door are changed. Note that time_to_open door is used to tell the control node when the door being monitored has been open too long.

DFD 4 Run Automated System - Start_Stop PAT			**Data Processes**	
stop_train_request	speed_request	target_speed_reached	4.1	4.2
TRUE	X	X	ON	X
X	INCREASE	X	ON	X
X	REDUCE	X	ON	X
X	STOP	X	ON	X
X	NONE	X	OFF	X
X	X	TRUE	OFF	X

Figure 20 - DFD 4 Table: Start_Stop PAT

Figure 20 uses a PAT to show which data processes are activated when a speed-change request is made (including stop requests). Note that target_speed_reached is used to tell the control node that a previous speed-change request is no longer needed.

3.3: CSCI External Interface Requirements:

3.3.1: Interface Identification and Diagrams:

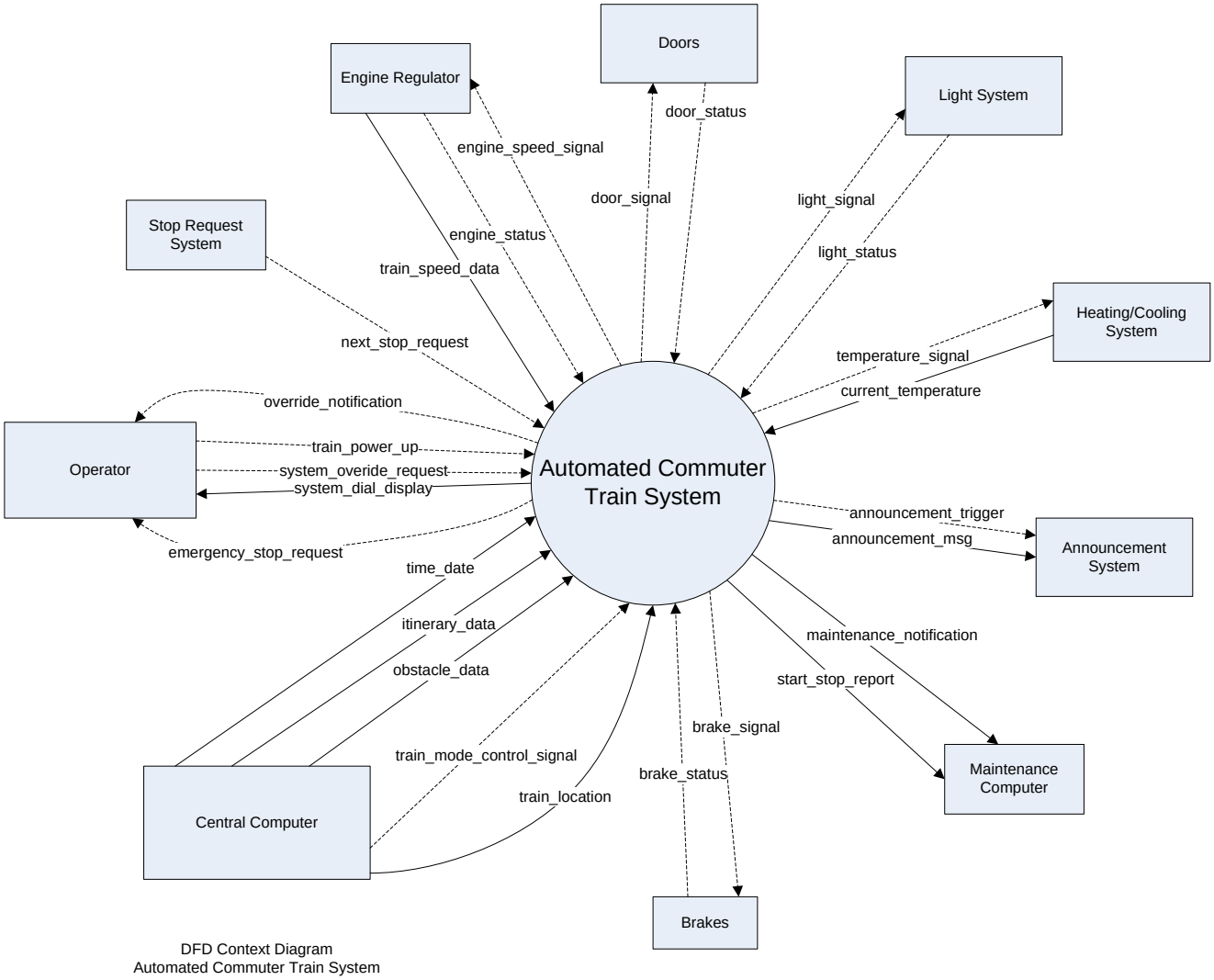


Figure 21 - DFD Context Diagram Automated Commuter Train System

3.3.2: External Interface Data Definitions:

Doors

Door Signal- Inputted as an a control input from the system to check if door has an obstruction

Door Status- Is a control output read into the system to check the status of the door if any obstruction are present

1. door _status(control,primitive)
    a. [“OPEN”||”CLOSED”||”OBSTRUCTED”]
2. door_signal(control, primitive)
    a. [“OPEN”||”CLOSE”]

Light System

Light signal- Inputted as an a control input from the system to tell lights to turn ON or OFF

Light status- Is a control output sent to the system to let the system know whether the lights are ON or OFF.

1. light_signal(control,primitive)
    a. [“ON”||”OFF”]
2. light_status(control,primitive)
    a. [“ON”||”OFF”]

Heating/Cooling System

Temperature signal- Inputted as an a control input from the system to determine if there needs to be changed to AC, HEAT, or if no change is needed, NONE.

Current temperature- is a data output sent to the system know and integer value of the current temperature inside the train.

1. current_temperature(data,primitive)
    a. //integer value of temperature inside train(^{o}F)
2. temperature_signal(control,primitive)
    a. [“AC”||”HEAT”||”NONE”]

Announcement System

Announcement msg- is a control output sent to the PA system to let passengers know; t ime_date, train_location, and itinerary_data.

Announcement trigger- is a control output that enables the PA system to be ON or OFF

1. announcement_msg(data,compound)
    a. =time_date + train_location + itinerary_data
2. announcement_trigger(control,primitive)
    a. [“ON”||”OFF”]

Maintenance Computer

Maintenance notification- is a data output from the system to the maintenance computer to notify of engine_break_failure , door_temp_light_failure

Start stop report- is a data output from the system after a train’s run to output start and stop data to the maintenance computer

1. maintenance_notificaiton(data,compoun d)
    a. =engine_brake_failure + door_temp_light_failure
2. start_stop_report(data,primitive)
    a. //after a train’s run, the ‘start_stop_data’ is compiled into this format to be sent to the maintenance computer.

Brakes

Brake signal- is a control output from the system to signal which type of brakes to enable:PHASE 1, PHASE2, SAFTEY, or EMERGENCY

Brake status- is control input read into the systems from the brakes letting the system know is brakes are OK or FAILED

1. brake_signal(control, primitive)
    a. [“PHASE1”||”PHASE2”||”SAFETY”||”EMERGENCY”]
2. brake_status(control,primitive)
    a. [“OK”||”FAILED”]

Central Computer

Train location- is data input read into the system with a value indicating the trains position

Train mode control signal- is a control input read into the system lets the system know LOCAL, EXPRESS, REQUEST ONLY

Time date- is a data input read into the system to verify time and date

Obstacle data- is a data input read into the system letting it know if the tracks are CLEAR or BLOCK

Itinerary data- is a data input read into the system that the system queues to know where to go and what mode to operate in

1. train_location(data,primitive)
    a. //value indicating if train's position is at a valid boarding location
2. train_mode_control_signal(control,primitive)
    a. ["LOCAL"||"EXPRESS"||"REQUEST_ONLY"]
3. time_date (data,compound)
    a. = [time , date]
4. obstacle_data(data,primitive)
    a. ["CLEAR"||"BLOCK"]
5. itinerary_data(data,primitive)
    a. //queue destinations the train must travel

Operator

Emergency stop request- is a control output to the operator in case of a emergency due to a system failure

Override notification- is a data input to the system that the operator chooses either TRUE or FALSE

System override request- is a control output from the operator to the system either TRUE or FALSE

System dial display- is a data output from the system to the operator a status display

Train power up- is a control input into the system that verifies if train power up is equalto TRUE or FALSE

1. emergency_stop_request(control,primitive)
    a. //signal sent to the operator requesting an emergency stop due to a system failure
2. override_notification(control,primitive)
    a. ["TRUE"||"FALSE"]
3. system_override_request(control, primitive)
    a. ["TRUE"||"FALSE"]
4. system_dial_display(data,compound)
    a. //data sent to operator dials containing information about the following: door obstructions, light status, train speed, train temperature, stop requests
5. train_power_up(control,primitive)
    a. ["TRUE"||"FALSE"]

Stop request system

Nest stop request- is a control output letting the system know that a stop request is requested

1. next_stop_request(control,primitive)
    a. //control signal from passenger requesting a stop at the next location

Engine regulator

Engine speed signal- is a control output from the system to let the engine regulator know to accelerate

Engine status- is a control input letting the system know if engine failed by equaling FAILED or OK if not failed

Train speed data- is a data input letting the system know the current value of the speed

1. engine_speed_signal(control,primitive)
    a. //signal sent to engine to accelerate
2. engine_status(control,primitive)
    a. [“OK”||”FAILED”]
3. train_speed_data(data,primitive)
    a. //value corresponding to the current speed of the train

3.4: CSCI Interface Requirements:

3.4.1: P-specs for Primitive of Monitor Train Status

Name:
1.1
Title:
Determine Which Sensor Failed

Input/Output:
sensor_ID: data_ in
sensor_failure: control_out

Body:
Sensor_ID is read in and from there it determines if there is a sensor_failure and outputs: senor_ID= 0 so sensor_failure=NONE, senor_ID= -1 so sensor_failure=DOOR, senor_ID= -2 so sensor_failure=LIGHT, senor_ID= -3 so sensor_failure=TEMP.

Name:
1.2
Title:
Send Maintenance Notification

Input/Output:
maintenance_notification: data_out
note_sent: control_out

Body:
Based on what activated the process a notification will be sent out if any of these =FAILED: DOOR, TEMP, LIGHT, ENGINE, BRAKE. After a notification has been sent note_sent in sent out to Status PAT to deactivate the notification.

3.4.2: P-specs for Primitive Functions of Monitor On-train System Sensors

NAME:
2.1

TITLE:
Determine Valid Sensor Readings

INPUT/OUTPUT:
current_temperature: data_in
train_speed_data: data_in
light_data: data_in
sensor_ID: data_out
sensor_status: control_out

BODY:
The input data (current_temperature, train_speed_data, and light_data) will be polled every one second to determine that valid sensor readings are being assessed. As long as the sensors are operating correctly, the outputs will remain unchanged as sensor_status is set to "OK" and sensor_ID is set to "0". In the event of a sensor failure, sensor_status is set to "FAILED" and sensor_ID is set to "-1," "-2," or "-3" to indicate which sensor has failed.

NAME:
2.2

TITLE:
Maintain Lights

INPUT/OUTPUT:
time_date: data_in
target_light_status: control_out

BODY:
The data input, time_date, is used to determine whether the target_light_status should be set to "ON" or "OFF". The date is used in conjunction with the time to designate the appropriate time intervals for the train lights to be turned on.

NAME:
2.3

TITLE:
Determine Target Temperature

INPUT/OUTPUT:
time_date: data_in
temperature_parameters: data_in (store)
temperature_range: data_out

BODY:

The time_date input provides the date and time to be used in determining the target temperature for the current season. The temperature_parameters data store is accessed by this function to retrieve the correct temperature range to be outputted to temperature_range.

NAME:
2.4

TITLE:
Change/Display Temperature

INPUT/OUTPUT:
temperature_range: data_in
current_temperature: data_in
temp_change_request: control_out
current_temperature: data_out

BODY:
If the current_temperature input is higher than the maximum value of the temperature_range input then the temp_change_request output is set to “LOWER” and the current_temperature input is passed on as an output. If the current_temperature input is lower than the minimum value of the temperature_range input then the temp_change_request output is set to “RAISE” and the current_temperature input is passed on as an output. If the current_temperature input is within range of the temperature_range input then the temp_change_request output is set to “NONE” and the current_temperature input is passed on as an output.

NAME:
2.5

TITLE:
Determine Speed Change

INPUT/OUTPUT:
train_speed_data: data_in
current_train_speed: data_out
speed_change_request: control_out

BODY:
The train_speed_data input is broken up into the current speed of the train and the target speed of the train. The current speed is passed on as an output through current_train_speed. If the current speed is higher than the target speed, then the speed_change_request output is set to “LOWER.” If the current speed is higher than the target speed, then the speed_change_request output is set to “RAISE.” If the current speed is the same as the target speed, then the speed_change_request output is set to “NONE.”

NAME:
2.6

TITLE:
Generate Operator Dial

INPUT/OUTPUT:
light_data: data_in
current_train_speed: data_in
current_temperature: data_in
system_dial_display: data_out

BODY:
The light_data, current_train_speed, and current_temperature inputs are passed to this function to be organized in a form that can be interfaced with the Train Operator's dial display. The system_dial_display output contains all of the sensor data to be sent to the dial display.

3.4.3: P-specs for Primitive of Schedule Train

Name:
3.1
Title:
Process Itinerary and Obstacle Data

Input/Output:
obstacle_data: data_in
itinerary_data: data_out
itinerary_data: data_in
train_obstacle_data: data_out
obstacle_present: control_out

Body:
If Obstacle_data =0, then obstacle_present control output is set to FALSE. Otherwise obstacle_data is not equal to 0 then obstacle_present equals TRUE and the obstacle_data is passed as an output through train_obstacle_data. Ititeneray in passed as a data input and if obstacle_present=False, itinerary is then passed as data out.

Name:
3.2
Title:
Store Start_Stop Data and Activate PA System

Input/Output:
time_to_stop: control_out
train_location: data_out
announcement_msg: data_out
start_stop_data: data_out
train_location: data_in
time_date: data_in
start_stop_train: control_out

Body:
Itinerary data is passed as an input to determine the value of start_stop_train. Time_date is passed as an input to let start_stop date know when it needs to start or stop. Train_location is passed as an input to let time_to_stop to activate it's control output. Announcement_msg is passed as an output. Train_location passed as an output to determine where at on the track it is located. Start_stop_data is passed out to a data store to record every start and stop that train.

Name:
3.3
Title:
Send Start_Stop Data to Maintenance Computer

Input/Output:
start_stop_data: data_in(store)
start-stop_report: data_out

Body:
Start_stop_data is read in from the start_stop database so start_stop_report will be passed out to the maintenance computer database at the end of trains run.

3.4.4: P-specs for Primitive Functions of Run Automated System

NAME:
4.1

TITLE:
Manage Doors

INPUT/OUTPUT:
train_location: data_in
door_signal: control_out
announcement_trigger: control_out
time_to_open_door: control_out

BODY:
If the Manage Doors process is activated, then the door_signal output is set to “OPEN,” and the announcement_trigger is set to “ON.” After a designated time (however long the doors are to remain open) the door_signal is set to “CLOSE.” If the train_location input is equal to a valid boarding location, then the time_to_open_door control output is set to “TRUE.”

NAME:
4.2

TITLE:
Manage Train Speed

INPUT/OUTPUT:
obstacle_data: data_in
brake_signal: control_out
engine_speed_signal: control_out
target_speed_reached: control_out

BODY:
If a stop request is made, then the Manage Train Speed process is activated and the brake_signal control output is set to “PHASE2.” If the obstacle_data input is equal to “BLOCK” then the brake signal control output is set to “PHASE2.” If a raise speed request is made, then the Manage Train Speed process is activated and the engine_speed_signal control output is sent to increase speed. If a lower speed request is made, then the Manage Train Speed process is activated and the brake_signal control output is set to “PHASE1.” When the target speed is reached for any situation (stopped, lowered speed, or raised speed), the target_speed_reached control output is set to “TRUE.”

3.5: CSCS Internal Data Requirements

- **start_stop Database:**
  - **Purpose**: Stores the time and date for each train's start and stop location. After the train completes a run, this data is compiled and sent to the central computer. (DFD 3.3)
  - **Location**: DFD 3-Schedule Train
  - **Inputs/outputs**: start_stop_data
  - **Data Definition**: start_stop_data = time_date + train_location
- **temperature_parameters:**
  - **Purpose:** Predefined read-only parameters which are used to determine the temperature_range in accordance to the time_date.
  - **Location:** DFD 2-Moniter On-train Status
  - **Outputs/Data Definition:** temperature_parameters //List of temperature ranges (unique depending on the time and date)

3.6: Data Dictionary Entries

1. announcement_msg(data,compound)
    a. =time_date + train_location + itinerary_data
2. announcement_trigger(control,primitive)
    a. [“ON”||”OFF”]
3. brake_signal(control, primitive)
    a. [“PHASE1”||”PHASE2”||”SAFETY”||”EMERGENCY”]
4. brake_status(control,primitive)
    a. [“OK”||”FAILED”]
5. current_temperature(data,primitive)
    a. //integer value of temperature inside train(^{o}F)
6. current_train_speed(data,primitive)
    a. //numerical value of the train’s speed which is sent to the operator’s dial
7. door _status(control,primitive)
    a. [“OPEN”||”CLOSED”||”OBSTRUCTED”]
8. door_signal(control, primitive)
    a. [“OPEN”||”CLOSE”]
9. door_temp_light_failure(control,PAT)
    a. [“LIGHT”||”DOOR”||”TEMP”]
10. emergency_stop_request(control,primitive)
    a. //signal sent to the operator requesting an emergency stop due to a system failure
11. enable(control,primitive)
    a. [“TRUE”||”FALSE”]
12. engine_brake_failure(control,PAT)
    a. [“ENGINE”||”BRAKE”]
13. engine_speed_signal(control,primitive)
    a. //signal sent to engine to accelerate
14. engine_status(control,primitive)
    a. [“OK”||”FAILED”]
15. itinerary_data(data,primitive)
    a. //queue destinations the train must travel
16. light_change_request(control,primitive)
    a. [“ON”||”OFF”]
17. light_signal(control,primitive)
    a. [“ON”||”OFF”]
18. light_status(control,primitive)
    a. [“ON”||”OFF”]

19. maintenance_notificaiton(data,compound)
    a. =engine_brake_failure + door_temp_light_failure
20. manual_mode_control(control,PAT)
    a. [“TRUE”||”FALSE”]
21. next_stop_request(control,primitive)
    a. //control signal from passenger requesting a stop at the next location
22. note_sent(control,PAT)
    a. [“ON”]
23. obstacle_data(data,primitive)
    a. [“CLEAR”||”BLOCK”]
24. obstacle_present(control,primitive)
    a. [“TRUE”||”FALSE”]
25. obstructed_door_notification(control,PAT)
    a. [“TRUE||”FALSE”]
26. override_notification(control,primitive)
    a. [“TRUE”||”FALSE”]
27. sensor_failure(control,primitive)
    a. [“NONE”||”LIGHT”||”DOOR”||”TEMPERATURE”]
28. sensor_ID(data,primitive)
    a. [0 || -1 || -2 || -3] //identification of failed sensor:
        i. 0-none, 1-temp, 2-speed, 3-light
29. sensor_status(control,primitive)
    a. [“OK”||”FAILED”]
30. speed_change_request(control,primitive)
    a. [“RAISE”||”LOWER”||”NONE"]
31. speed_request(control,PAT)
    a. [“INCREASE”||”REDUCE”||”STOP”||”NONE”]
32. start_request(control,primitive)
    a. [“TRUE”||”FALSE”]
33. start_stop_data(data store,compound)
    a. =time_date + train_location
34. start_stop_report(data,primitive)
    a. //after a train’s run, the ‘start_stop_data’ is compiled into this format to be sent to the maintenance computer.
35. start_stop_train(control,primitive)
    a. [“START”||”STOP”]
36. stop_request(control,primitive)
    a. [“TRUE”||”FALSE”]
37. stop_train_request(control,PAT)
    a. [“TRUE”]
38. system_dial_display(data,compound)

    a. //data sent to operator dials containing information about the following: door obstructions, light status, train speed, train temperature, stop requests.
39. system_override_request(control, primitive)
    a. [“TRUE”||”FALSE”]
40. systems_ok(control,primitive)
    a. [“TRUE”||”FALSE”]
41. target_light_status(control,primitive)
    a. [“ON”||”OFF]
42. target_speed_reached(control,PAT)
    a. [“TRUE”||”FALSE”]
43. temp_change_request(control,primitive)
    a. [“RAISE”||”LOWER”||”NONE”]
44. temperature_parameters(data store)
    a. //List of temperature ranges (unique depending on the time and date)
45. temperature_range(data,primitive)
    a. //determined range used to calculate if change is needed
46. temperature_signal(control,primitive)
    a. [“AC”||”HEAT”||”NONE”]
47. time_date (data,compound)
    a. = [time , date]
48. time_to_open_door(control,PAT)
    a. [“TRUE”||”FALSE”]
49. time_to_stop(control,primitive)
    a. [“TRUE”||”FALSE”]
50. train_location(data,primitive)
    a. //value indicating if train’s position is at a valid boarding location
51. train_mode_control_signal(control,primitive)
    a. [“LOCAL”||”EXPRESS”||”REQUEST_ONLY”]
52. train_power_up(control,primitive)
    a. [“TRUE”||”FALSE”]
53. train_ready_power_down(control,primitive)
    a. [“TRUE”||”FALSE”]
54. train_speed_data(data,compound)
    a. //value corresponding to the current speed and the target speed of the train