SGC - Siesta Gardens Controller

Software Architecture Document

SAD Version 1.3

Team 03: B-STACK
11 May 2021

Amun Kharel
Brandon Stringham (manager)
Cody Crane
Krista Conley
Shreeman Gautam
Tanner Hunt

CS 460 Software Engineering

Table of Contents

1 Introduction

"B-STACK" is a software design group consisting of Brandon Stringham, Shreeman Gautam, Tanner Hunt, Amun Kharel, Cody Crane, and Krista Conley. B-Stack's Software Architecture Document (SAD) provides a comprehensive architectural overview of the Siesta Gardens Controller software (SGC).

The purpose of this project is to design a theme park control system to track and transport guests from the main entrance of the key (East) to the T. rex exhibit (West) and back again with high safety protocols for the protection of the guests. The main entrance of the key will allow guests to access tokens which in turn allows access to the self- driving vehicles at the main entrance. Once in a vehicle, the guests will be transported to the T. rex exhibit and then back safely to the main entrance. There are security measures which have software and hardware implementation that will communicate two-ways with central control with state of the art technology to ensure the highest safety measures. Thus, the whole SGC system consists of the nine subcomponent controllers that are monitored by the Central Control system to ensure a safe, fun, and exciting vacation experience!

The purpose of this document is to map out the software and how the software is structured to be able to communicate between the software architect, B-STACK, and the customers, stockholders, and other developers who want to realize the project. In regards to the software architecture, the intended audience for this document is the developers for the final product. In essence, the purpose of this document is to link the idea and reality. This document will discuss the following topics:

- Section 2: Design overview: Design approach, Design diagram, External interfaces, Diagram explanation.
- Section 3: Component Specifications: Logical view of each specific component
- Section 4: Two Sample Use Cases: Processing and Illustrating the execution process
- Section 5: Design Constraints: Limitations on the system implementation

2 Design Overview

This system was designed with the central control as the centralized control point where park operators can view status information or request that components execute specific actions. Because of this, the central control directly communicates with the software controllers for each of the devices in the park and can get their status information or send them commands.

Each device has a software controller attached to it that monitors the status of the hardware device and is equipped to send messages to the central controller. Some of the controllers, such as the RFID scanners, RFID Bracelets, token kiosks, loudspeakers, cars, and the storage devices will also receive messages/commands from central control which they use to interact and control the associated hardware devices.

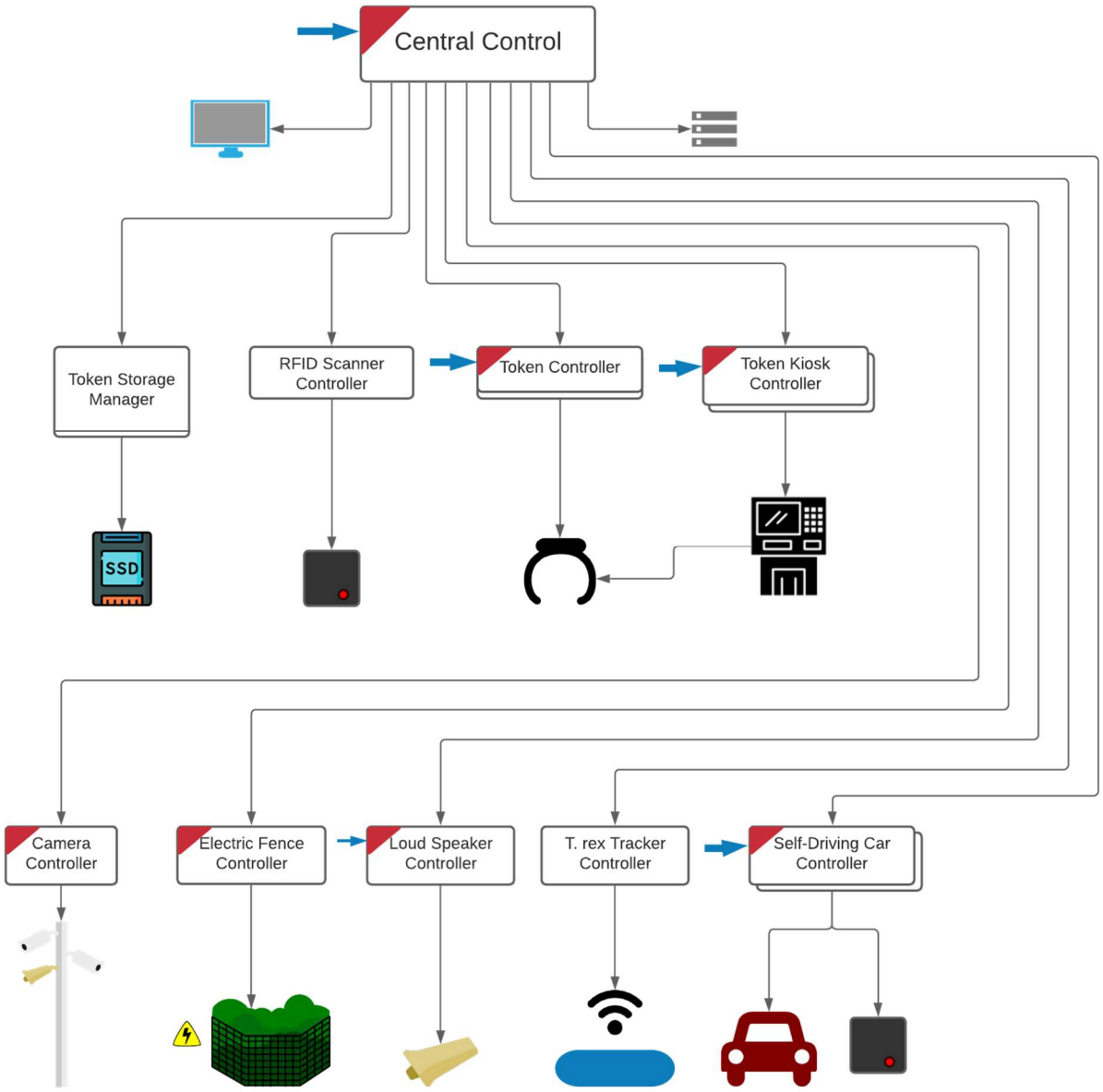

Figure 1: Software Architecture Design Diagram

In Figure 1, there is an overarching controlling component that starts the other components when the whole system powers on. This component communicates with nine subcomponent controllers. There is constant two-way communication between the subcomponent controllers and the central controller to ensure the safety of the guests

There are several external devices that the system connects to, the token kiosk with an output screen, the RFID token bracelets, the vehicles, the electronic fence, the cameras, the RFID scanners, GPS T. Rex monitor, the PA loudspeakers, and the Token Storage database. The token system will wait for input from the guest via a keypad and display messages via the output screen. The vehicles will have an RFID Scanner that will detect the guests via the token and proceed to the exhibit. The electric fence and camera devices send feedback to the central controller via radio transmitters and the loudspeakers wait for audio to play when appropriate. The RFID scanners located around the key pick up the bracelet signals to send back to central control with the location of the guests. The T. Rex GPS monitor sends the animal's vitals and location back to central control for health and location statuses.

3 Component Specifications

Each of the individual components in Figure 1 handle the logic related to a specific portion of the system operation. Some of these components only have to handle a single action, but several handle more complex actions to accomplish their component's goal. Together, the components make up the functionality of the whole system. Each subsection will go into detail outlining the method signatures for each of the Controllers from Figure 1.

There is a Message Parent class that defines a unified interface for passing around generic messages. Then there are a bunch of sub-classes that extend this Message class and have a certain MessageType associated with them, as well as any data they may need to pass in the message from one controller to another. The Message types for the internal queues are:

AUDIO_BROADCAST

This message type is used to send audio to be played.

DEVICE_STATUS_UPDATE

This message type is used to update the central controller of the status of other device statuses.

EMERGENCY_DEESCALATION

This message type is used to stand down the emergency components.

FENCE_STATUS

This type of message updates the central controller on the status of the fence.

GUEST_LOCATION

This type of message updates the central controller of the location of the guests.

PASSENGER_DEREGISTRATION

This message type is used to de-register the token bracelet after the guest has finished the ride.

PASSENGER_ENTERED

Used to notify the central controller that a passenger has boarded a vehicle. Contains the vehicle and passenger identifiers.

PASSENGER_EXITED

Used to notify the central controller that a passenger has exited a vehicle. Contains the vehicle and passenger identifiers.

PASSENGER_REGRISTRATION

This message type is used to register the token bracelet once the guest has received their token bracelet .

RECEIPTS_LIST

This message type is used to request the daily receipts list from the website.

UPDATE_TOKEN_DISPLAY

This message type is used to update the token bracelets.

TOKEN_INTIALIZED

This message type is used for initializing the token bracelet.

TOKEN_KIOSK_STATUS

The message type is used for sending the current status of each kiosk.

TOKEN_STATUS

The message type is used for sending the current status of each token bracelet to ensure it is working correctly.

TREX_STATUS

This message type is used to update the central controller to the status of the T rex monitor.

VEHICLE_STATUS

This message type is used to update the central controller of the status of vehicles

EMERGENCY_OVERRIDE

This message type is used to signal controllers to begin emergency operations.

EMERGENCY_HANDLED

This message type is used to signify the end of emergency operations

VEHICLE_INITIALIZATION

This message type is used to tell a vehicle to start up

OPERATOR_EMERGENCY_REQUEST

This message type is used to tell the user interface to alert the operator to an emergency situation and request their input.

TOKEN_INTIALIZED

This message type is used to tell the central controller that a token has been initialized by a kiosk

TOKEN_STATUS

This message type is used to update the central controller of the status of the bracelet

GUEST_LOCATION

This message type is used by the RFID scanners to tell the central controller what guests are in the vicinity of the RFID scanner

3.1 Central Controller

The Central Controller is the overarching component that monitors/commands each sub-component controller, displays the status to the operators, and accepts command requests from the park operators. Under normal operations, the controller will just update the status displays with the status of each component of the SGC system, and send commands from operators to those components. If the system or the park operators detect faults in the security system, it will enter emergency mode and send emergency overrides to subcomponent controllers of the system. For safety, the Central Controller will be online and monitored 24/7 by park staff along with the Electric Fence and T. Rex GPS chip regardless of park closure. The Central Controller class contains the following method signatures:

void run()

This is the main message processing loop that will wait for messages from the subcomponent controllers to arrive on an internal message queue.

void start()

Starts the whole SGC system up. It will download the list of purchased tokens for the day and store them in the token store from the third-party website. Brings the SGC sub-component controllers of the system online.

void processMessage(Message msg)

It parses, logs, and handles status messages received from the subcomponent controllers.

void processOperatorRequest(Message request)

This will process operator command requests and send appropriate commands to subcomponent controllers.

void updateDisplay(Message msg)

This displays the status of the components and all the guests/vehicles information and locations.

void startManualEvac(int operatorEnteredPassword)

This is a password protected function. If no password or an incorrect password is entered an evacuation is not started. If the correct password is entered, it will send out emergency override messages to the subcomponent controllers.

void startAutoEvac()

This is an automatic evacuation procedure in the event that the system detects a critical fault, a timer will begin and the central control operator will be prompted. If the timer reaches sixty seconds and has not received a not received a de-escalation request from the central control operator, it will send out emergency override messages to the component controllers.

void shutDownPark()

This is issued in the event of an unexpected closure for instance, when the T. rex needs medical attention. It will alert the guests that a closure is in effect and that they need to start evacuating. It will also send messages to the sub component controllers to start non-emergency evacuation. After all the guests have been evacuated, the non-essential sub-component controllers will be shut down.

void sendEmergencyHandled() & void sendEmergencyOverride()

These two functions send messages to the vehicles for handling emergency situations.

3.2 Token Controller

The Token Controller class contains the following method signatures:

void run()

This checks for incoming messages from the central controller and displays the message.

void sendHeartBeatStatus(Message msg)

Routinely sends a status update message to the central controller.

displayMessage()

Displays a communication message received on the token screen, and notifies the wearer via haptic feedback.

3.3 Token Kiosk Controller

The Kiosk Controller class contains the following method signatures:

void startUp()

Requests the list of valid receipts for the day from central control and starts waiting for user interactions or messages from central control.

void run()

This is the main processing loop that will check for incoming messages from the central controller and call the appropriate method based on the message type.

void sendHeartbeatStatus()

Sends a status update message with the status of the kiosk and the number of token bracelets remaining in the kiosk to the central controller.

int scan()

Method to scan a receipt and return a token ID for the project simulation.

int authenticateReceiptScan(int receiptNumber)

Check the guest's receipt number to ensure it is a valid receipt and if so, then dispense the token.

void dispenseToken(int newTokenID)

Load a bracelet from the bin, encode it with the guests token ID, and dispense the token bracelet to the user.

void sendHeartBeatStatus()

Sends Kiosk status update messages to the Central Controller

Map<int, int> requestTokenReceipts()

Requests the list of receipts for the day from the Central Control.

void shutdown()

Power off the kiosks

3.4 Self-Driving Car Controller

Each self driving car controller includes three threads of execution which will simultaneously handle incoming messages from the central controller, transitioning between and execute vehicle operations (Driving, Loading, Parking, etc.), and periodically sending the vehicle's heartbeat information to the central controller. The Self-Driving Car Controller class contains the following method signatures:

void run()

This is the main message processing loop that will wait for messages from the central controller to arrive on an internal message queue.

processMessage(Message message)

This parses and handles messages received from the central controller.

int[] readRFIDTokens()

The RFID scanner controlled by the car will read the list of signals sent from each token bracelet. It will send to central control an array of type Token of all the entered guests in that vehicle.

void startDrive(Location destination)

Lock vehicles doors and command the car to drive to a destination. Destination type is two integer numbers representing long and lat coordinates of the location.

void drive(Location destination)

Loops while the vehicle’s current location is not the destination. Calls arrival when the vehicle has arrived at its destination.

void arrival(Location location)

Unlock the doors and then based on the location, either start a viewing period timer, or wait for all guests to exit the vehicle and then wait to pick up new guests.

void alertGuests()

After a viewing period timer has expired, it will sound an alarm to alert guests to start loading now. It will also ask central control to send the guests' bracelets alert messages. Then it will begin loading the guests.

void loadGuests()

Enters the vehicle loading operational state, determines the loading procedure based on the vehicle's current location and emergency conditions, and then begins waiting for guests to get in the vehicle.

void sendHeartBeatStatus()

Periodically requests the location and status from the vehicle and sends them to the central controller.

Coordinates getLocation()

Get the current GPS location of the vehicle.Location is similar to the Destination type in that it is two integers representing long and lat coordinates of its location.

voidgetVehicleStatus()

Get the current vehicle status i.e. Okay, Broken.

void startEmergency()

Change from normal vehicle operations to emergency vehicle operations. It receives a MessageHandler from central control in this event.

3.5 Camera Controller

The Camera Controller class contains the following method signatures:

void run()

This is the main refresh loop that sends the latest frames of the feed to the central Controller.

3.6 Electric Fence Controller

The Electric Fence Controller class contains the following method signatures:

void sendHeartBeatStatus()

Send the Electric Fence status to the central controller periodically to ensure the devices are functioning correctly.

void checkFenceStatus()

Check the voltage nodes along the fence for voltage drops.

3.7 Loudspeaker Controller

The Loud Speaker Controller class contains the following method signature:

receiveAudio(String audioFilePath)

Receives audio files from central control. It will convert these files to electrical signals to play on the speakers for alerting or sending messages to the guests.

3.8 RFID Scanner Controller

The RFID Scanner Controller class contains the following method signatures:

run()

This is the main processing loop that checks for audio broadcast requests from the central Controller and sends back the list of currently connected RFID tokens.

void scanRFID()

Reads the signals of the guest RFID tokens and sends the location of the guests to central control for tracking.

voidsendHeartBeatStatus()

Periodically Sends constant heartbeat status to central control to ensure they are functioning correctly.

3.9 T. Rex Tracker Controller

The T. Rex Tracker Controller class contains the following method signatures:

void sendStatus()

Periodically sends the status of the vitals and location of the T. Rex to central control for health and tracking of the animal.

3.10 Token Storage Manager

The Token Storage Manager class contains the following method signatures:

void storeId(int tokenID)

Stores active token ID to the token data store.

void registerTokenIdToVehicle(int tokenID, String vehicleID)

Registers a token ID to a vehicle.

void removeId(int tokenID)

Remove and invalidate an active ID from the data store.

void deregisterPassenger(String vehicleID, int tokenID)

Deregistering a token ID from a vehicle.

4 Sample Use Cases

This section outlines two major use cases for the system, a user receiving a token and getting to ride around and see the dinosaur, and a control center operator handling a fence failure emergency. These demonstrate the normal operation of the park, as well as how it responds in an emergency.

4.1 Token Kiosk Use Case

Use case: Guest Token Bracelet Dispensation

Primary Actor: Park Guest

Goal in context: Guest receives a Token Bracelet that has been programmed and registered to the SGC.

Preconditions: The Token Kiosk is idling, waiting for a guest to scan their pre-purchased Token receipt.

Trigger: Guest Scans their pre-purchased Token receipt

Scenario:

1. Guests observe the Token Kiosk.
2. Guest Scans their pre-purchased Token receipt.
3. Token Bracelet is programmed/registered.
4. Token bracelets are dispensed from the Kiosk.
5. Guest retrieves the newly dispensed token Bracelet.

Exceptions:

- If the scanned receipt is deemed invalid, the guest will be notified and no token bracelets will be dispensed.
- If a valid receipt is scanned and the Kiosk does not have any token bracelets left to give, the guest is directed to use another kiosk via the Token Kiosks' display.

4.2 Central Control Use Case

Use case: Fence Failure Park Emergency

Primary Actor: Park Central Control Operator

Goal in context: The park operator in central control is alerted that the electric fence has gone offline and validates the data sources and then begins an emergency evacuation.

Preconditions: The Central Controller is idly monitoring the park and receiving updates from all components

Trigger: The electric fence stops sending signals to the Central Controller

Scenario:

1. The park operator is observing the status of all the park components in the central control building.
2. The electric fence has some sort of fault and goes offline and stops sending signals to the Central Controller.

3. The Central Controller alerts the park operator that the fence is offline and prompts them to interact with the situation within sixty seconds.
4. The operator visually confirms the status of the component in the component status tables in the control center.
5. The operator checks the security camera footage and sees a breach in the fence.
6. The operator clicks the button to trigger the park's emergency mode and start park evacuation.
7. The Central Controller coordinates with all of the components to get all of the guests safely out of the park.

Exceptions:

- If the park operator inspects the camera footage and does not see any breach, and can then confirm with park security via walkie-talkie that the fence is still electrically charged, the operator will de-escalate the emergency situation and resume normal park operation after notifying maintenance that the fence sensors need maintenance.
- If the park operator does not manually intervene in the emergency alert within sixty seconds, the Central Controller will automatically initiate the emergency evacuation protocol.

5 Design Constraints

The system has limitations that are enforced on it by the constraints of the system. Some of these constraints are caused by ethical concerns, some by nature, some by engineering and physical space, and others by the demands of the guests.

5.1 Operating Schedule

The park schedule is constrained by the park's need for daylight and occasional maintenance. Because the dinosaur is only visible during the day and we do not want to disrupt its sleep schedule with night lights, the park schedule will be limited from sunrise to sunset. Since

the park will need routine maintenance and occasional software updates, the park will only be open from Monday to Saturday. On Sunday, park staff can conduct any necessary maintenance.

5.2 Severe Weather

The park's schedule and equipment are constrained by severe weather. The park will be closed to the public during major weather events, such as a hurricane, to protect the public. However, the SGC control system will still need to operate all security and emergency features to maintain the dinosaur enclosure. This requires the use of surge protectors on all circuits, back-up power generators to run the system, and automatic reboots if the system goes offline. The communication wires between subcomponents and the central control building also need to be insulated cable buried underground to protect from the weather. The electrical fence also needs to be designed in a way that it will not be adversely affected by rain. The park will remain open during normal inclement weather such as rain or mild wind. These protections help ensure that the park is fully operational at all times for the safety of the park staff and guests.

5.3 Guest Privacy

The system is required to track guests using cameras and RFID bracelets while they are at the park. To respect guest privacy, after three months, the aggregated data of the guests will be purged from the system. In that sense, the system is constrained by guest privacy to not hold guest information like name, camera footage and receipt information after three months.

5.4 Guest Tracking

The system is constrained to have passenger cars, as opposed to letting guests roam on foot, so that they are accounted for while they are in transit to the exhibit, so that they can easily escape in the event of an emergency and so that their time is not wasted and they don't risk getting tired while walking to the exhibit.

5.5 Natural Habitat

Some of the construction of the island is constrained by wanting to keep the feel of a natural habitat and safari for the guests and the dinosaur. This is because the self-driving cars will only follow one predefined road to the enclosure so they do not tear up the terrain and degrade the experience for the guests.

The enclosure design also needs to look natural for the dinosaur, therefore the security cameras inside the exhibit need to be disguised as a part of the local fauna. The exhibit will need to have many lines of visibility so the guests get the best chance of viewing the T. rex, but the enclosure needs to retain a natural look and so there may still be parts of the enclosure that guests cannot see.

5.6 Car Limitations

Due to the limited amount of space on the key, the number of cars is constrained to be ten vehicles that the system can use. For the safety of the park guests, five of these cars must be kept in reserve as spare cars in the parking lot near the enclosure to evacuate guests in the case of an enclosure failure. Each car can hold up to ten people.

5.7 Emergency Automation

Emergency management is constrained such that it cannot trigger an emergency response unless several conditions are met to prevent needless guest hysteria. In the case of a single system failure like the gate or GPS, the emergency management module will alert the control center staff, but it cannot immediately start evacuation procedures without their approval in case it is a sensor malfunction. If however, the staff do not respond to the alert within one minute, the system can begin emergency procedures in case the staff is somehow incapacitated.

Due to the short response time required, there must always be at least two staff in the control center at a time.

5.8 Automated Status Limitation

Not all systems have the capabilities to send constant heartbeat updates to the controllers, thus these systems will need to be manually checked during weekly maintenance. The announcement speakers need to be manually checked each week since they are regular speakers and can only play audio without being able to send back device health updates.

5.9 Animal Welfare

Due to the Animal Welfare Act, there exists a constraint such that the welfare of the animal must be kept in high regards.[1] The habitat will be cleaned and if for some reason the animal needs medical attention then the park will unexpectedly close down but the park will issue refunds to the affected guests. The Florida wildlife commission regulates the cleanliness and the humane treatment of wildlife. [2] Therefore, unexpected inspections may occur and the park will have to shut down temporarily to abide by state laws.

6 References

[1] “Animal Welfare Act,” *NAL*. [Online]. Available: https://www.nal.usda.gov/awic/animal-welfare-act. [Accessed: 09-Apr-2021].

[2] “FWC Overview,” *Florida Fish And Wildlife Conservation Commission*. [Online]. Available: https://myfwc.com/about/overview/. [Accessed: 09-Apr-2021].