Becker, Nathan

[Email address]

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

countess Quanta Architecture

[Document subtitle]

# Layer Diagram



The idea is to create an architecture that contains isolated modules with common interfaces, to allow for independent testing and the ability to switch out hardware modules with minimal code changes. Each green and blue square should be in its own independent class.

# Boot Sequence



This sequence will be done every time the robot is turned on.

# State Machine



This is a simplified version of the robot’s state machine. It monitors its inputs, waiting for a decision, and then it executes an action based on the decision and moves on to the next decision. If fuzzy logic is being used, and no option evaluates high enough, it will search through previous decisions to see if there is a better option available.

# Decision Initializer



This is the logic flow for loading the decision information from the database into the objects. This is an essential step, as it allows for better performance. It essentially looks at the database for any related rows in the Option tables to the current decision, and copies the values into the correct locations inside C# object collections. It looks for the “ParentID” field in objects and uses that to construct a tree of available options and sub-options to evaluate. The object without a ParentID field is the root object.

# Decision Hierarchy

This is an example of a decision and the options that make up the decision. Each logic node is a separate “Option” row, with the appropriate ParentID field that will enable the decision engine to construct a tree of objects with which it will then evaluate.

# Making a Decision



This is the logic flow of making a decision. It looks at the decision object, and evaluates each option/sub-option and comes up with a fuzzy logic value for each option. The option that contains the highest value is used.

# Decision Interface



This is the interface design for the decision engine. The parentEvaluators will contain lists of type “IEvaluator”. Each of those evaluator objects will be initialized depending on what type of evaluator they are – if button evaluators, then the name of the button will be given during construction of the object, if a gesture, then a gesture name, etc.

# Action Example



This is an example of an action. Each action can be made up of sub-actions, each of which can be targeting different modules. They can be run in sequence, parallel, or both.

# Retrieving Actions



This is the logic flow for retrieving actions. It looks and reconstructs a tree based on the “ParentActionID” field – the row with no ParentActionID field is the root entry.

# Running Actions



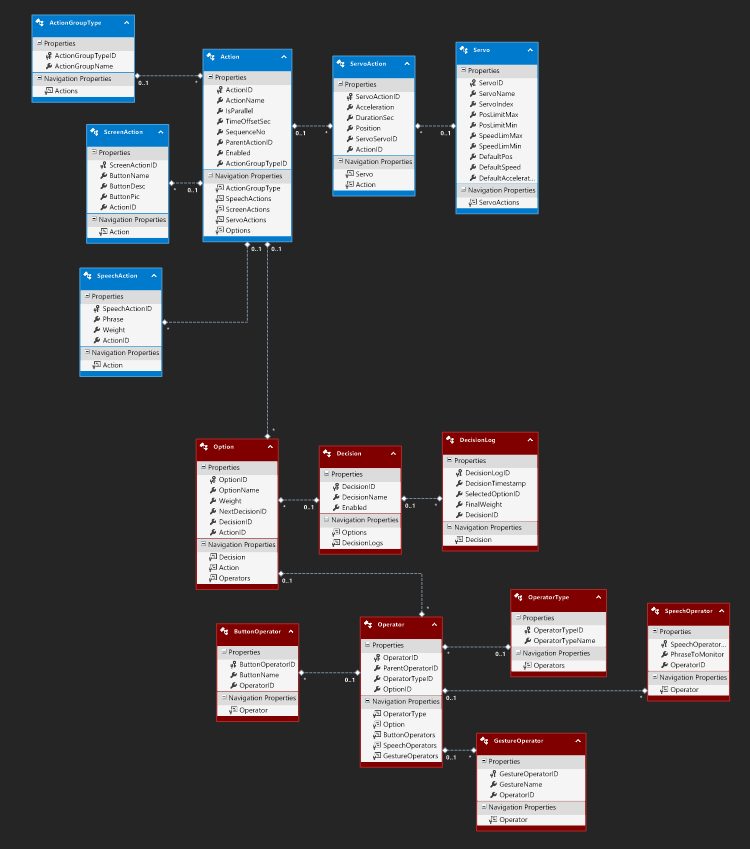
This is the logic flow that dictates how the actions are run – it looks at the main object and launches either sequential or parallel processes or both, depending on what the object specifies.

# IAction Interface



This is the interface and associated objects for the Action engine. There will be a root IAction, which will contain child IActions specified in the ParentAction object. Because we’re using an interface, each action can do something unique, like turn a servo or say a phrase or display a grid of buttons on a screen. Their abilities will be initialized when they’re initially constructed using values from the database.

# Database



Here’s a big picture view of the database. The interactive version can be found in the only .edmx file in the project. It uses Entity Framework, which eliminates the need to use SQL to query and store data in the database. Currently, the database schema has been created, but an actual database file with seeded values has not. This can be done according to the online documentation (listed at the end of the report), and there are built-in tools for interacting with the local database file. In the diagram, the blue tables are the Action tables and the red tables are the Decision tables. Following is a description of each table.

Note: The top value in each table is a primary key identity field, so it automatically adds a unique integer ID to each new value that is added. When adding new rows, this field should be ignored, as it is handled automatically during the insertion process.

# ActionType Table

* Specifies what kind of action the associated action (specified by “ActionID” is
  + “ParentAction” – Says that the action is a parent of one or more sub-actions
  + “ScreenAction” – Says that the action is linked to a screen activity
  + “SpeechAction” – Says that the action is linked to a speech activity
  + “ServoAction” – Says that the action is linked to a servo activity

# ScreenAction Table

* “ButtonName” is the name of the button displayed on the screen
* “ButtonDesc” is a description of the button
* “ButtonPic” is a picture displayed on the button

# SpeechAction Table

* “Phrase” – the phrase to say
* “Weight” – allows the ability for a particular phrase to come up more/less often when ran through a randomizer algorithm

# Servo Table

* “ServoName” – the name of the servo
* “ServoIndex” – the Polulu servo ID
* “PosLimitMax” – the maximum position of the servo
* “PosLimitMin” – The minimum position of the servo
* “SpeedLimMax” – the maximum speed of the servo
* “SpeedLimMin” – the minimum speed of the servo
* “DefaultPos” – the default position of the servo
* “DefaultSpeed” – the default speed of the servo
* “DefaultAcceleration” – the default acceleration of the servo

# ServoAction

* “Acceleration” – specifies how fast the servo accelerates
* “DurationSec” – specifies how long the servo is running
* “Position” – specifies the ending position of the servo

# Action

* “ActionName” – The name of the action
* “IsParallel” – specifies whether or not the action should be executed in parallel
* “TimeOffsetSec” – how long to wait before firing the action
* “SequenceNo” – the sequence number of the action, if the action is not parallel
* “Enabled” – Specifies whether or not the action is enable – this allows for actions to be “deleted” without really deleting the actions or sub-actions, in the event the action needs to be re-enabled.

# Option

* “OptionName” – the name of the option
* “Weight” – the fuzzy logic weight assigned to the option
* “NextDecisionID” – the ID of the decision to run if this option is chosen
* “DecisionID” – the ID of the associated decision
* “ActionID” – the ID of the action to run if the option is chosen

# Decision

* “DecisionName” – the name of the decision (for human readability)
* “Enabled” – specifies whether or not the decision is enabled.

# DecisionLog

* This keeps a log of each decision that was made
* “DecisionTimestamp” – The timestamp when the decision was made
* “SelectedOptionID” – The option that was chosen
* “FinalWeight” – The fuzzy logic weight of the final decision
* “DecisionID” – the associated decisionID

# OperatorType

* OperatorTypeName – Name that determines what type the associated operator is
  + “ParentOperator” – says that there are children operators
  + “GestureOperator” – says that the operator is based on a gesture
  + “SpeechOperator” – says that the operator is based on a speech phrase
  + “ButtonOperator” – says that the operator is based on a button action
* More options can be added, but additional tables will need to be added as well.

# Operator

* ParentOperatorID – the ID of the parent operator option
* OptionID – the ID of the associated option
* Weight – the fuzzy logic weight of the operator
* LogicGate – only useful for operators of the “ParentOperator” type – specifies how to evaluate children operators – options are “AND”, “OR”, “NAND”, “NOR”, or “XOR”

# ButtonOperator

* ButtonName – the name of the button to monitor – if the button with the matching name is pushed, the operator returns “true”.

# GestureOperator

* GestureName – the name of the gesture to monitor

# SpeechOperator

* The name of the phrase to monitor

# Resources

* Subscribing to events (<http://msdn.microsoft.com/en-us/library/ms366768.aspx> )
* Interfaces (<http://msdn.microsoft.com/en-us/library/ms173156.aspx> )
* Entity Framework (<http://msdn.microsoft.com/en-us/data/ee712907> )
* Task Parallel Library (<http://msdn.microsoft.com/en-us/library/dd460717(v=vs.110).aspx> )