Carl Lange & Ciarán McCann
C00138612 & C00134748
February 17, 2010

SYSTEM DESIGN DOCUMENT - SOLOMON (ROBOCODE)

Introduction

Solomon is an Artificial Intelligence robot, whose main feature is the ability to choose a tactic based on its situation. Solomon has a library of approximately 16 tactics, (the number of which can be easily added to), and will pick the best tactic for its current situation. Solomon was coded and designed in fewer than ten days in January 2010.

It's based off of the ideas of a finite state machine (http://en.wikipedia.org/wiki/Finite-state_machine).

The AI assesses its health and chooses which state it's in, i.e extremely aggressive, aggressive, defensive, or extremely defensive. These states have certain behaviours (henceforth called tactics) associated with them. These tactics are used, assessed for efficiency, and used again based on their past efficiency.

Essentially, this means that we can have as many different behaviours as we like, and Solomon will pick the best one for the situation.

Tactics Library

Because of the modularity in tactics, it's very easy to add or remove tactics from the robot. This means that tactics can easily be tested and migrated into the library.

Every tactic inherits from the CTactics class, and follows a simple naming convention:

Tactic Library	**0**	**1**	**2**	**3**
State 1 (extremely aggressive)	CTactic_ea0	CTactic_ea1	CTactic_ea2	CTactic_ea3
State 2 (aggressive)	CTactic_a0	CTactic_a1	CTactic_a2	CTactic_a3
State 3 (defensive)	CTactic_d0	CTactic_d1	CTactic_d2	CTactic_d3
State 4 (extremely defensive)	CTactic_ed0	CTactic_ed1	CTactic_ed2	CTactic_ed3

The tactic library is, a two-dimensional array. It's filled (manually) with the tactics, during Solomon's construction (that is, when a match starts). Each tactic in the library overrides several methods in the

CTactic class, such as run_(), onScannedRobot_() etc. These methods have underscores behind their names as they otherwise would share a name with methods in Solomon's core, and we'd like to avoid confusion.

We're trying to keep each tactic simple-ish, as we only have a few days to code the entire AI.

//TODO: List of tactics. Yes, I know that's a lot of writing, you woman.

The Core

Solomon's core, that is, gauging and switching tactics, was mostly programmed by Ciarán.

//TODO: Finish this.

CTactic Class

The CTactic class, which is the superclass for every tactic, contains some methods, which are described in detail here.

run_(solomon s)

This is an empty method, put here for overriding. Takes in Solomon so that it can, if necessary, call methods only available to Solomon. This can be said about almost all methods hence.

onScannedRobot_(solomon s, ScannedRobotEvent e)

onHitByBullet_(solomon s, ScannedRobotEvent e)

Same as run_(solomon s).

fire(solomon s, double enemyDist)

This is a dynamic firing method. It modulates the power of the bullet to be fired (between 0.1 and 3.0, the global limits). The power is calculated by two things: the enemy distance and the bias, which is decided (through a case statement) by the status of the robot (aggressive, defensive, et cetera). If Solomon's in extremely aggressive mode, it'll be much more likely to fire with full strength, and if it's in extremely defensive mode, it will almost never fire with full strength.

isGoodTactic(int status)

This is called by pickTactic in the AI class. It adds up every item in the gaugingList (which is an ArrayList of bytes), divides the total by the size of the list, and multiplies that by 100. Then, it checks whether this result is greater than the gauging threshold. If it is, it returns true (i.e. That the tactic is good). If not, the tactic is not good, and returns false.

getRandom()

Simply returns a random number. Done here to combat code duplication.

getRandom(int n)

Returns a random number between zero and n.

[There are also a few simple radian translations of calculations that would otherwise return degrees. This is done for compatibility with almost every method in the native Math class.]

//TODO: Pad this out a little more.

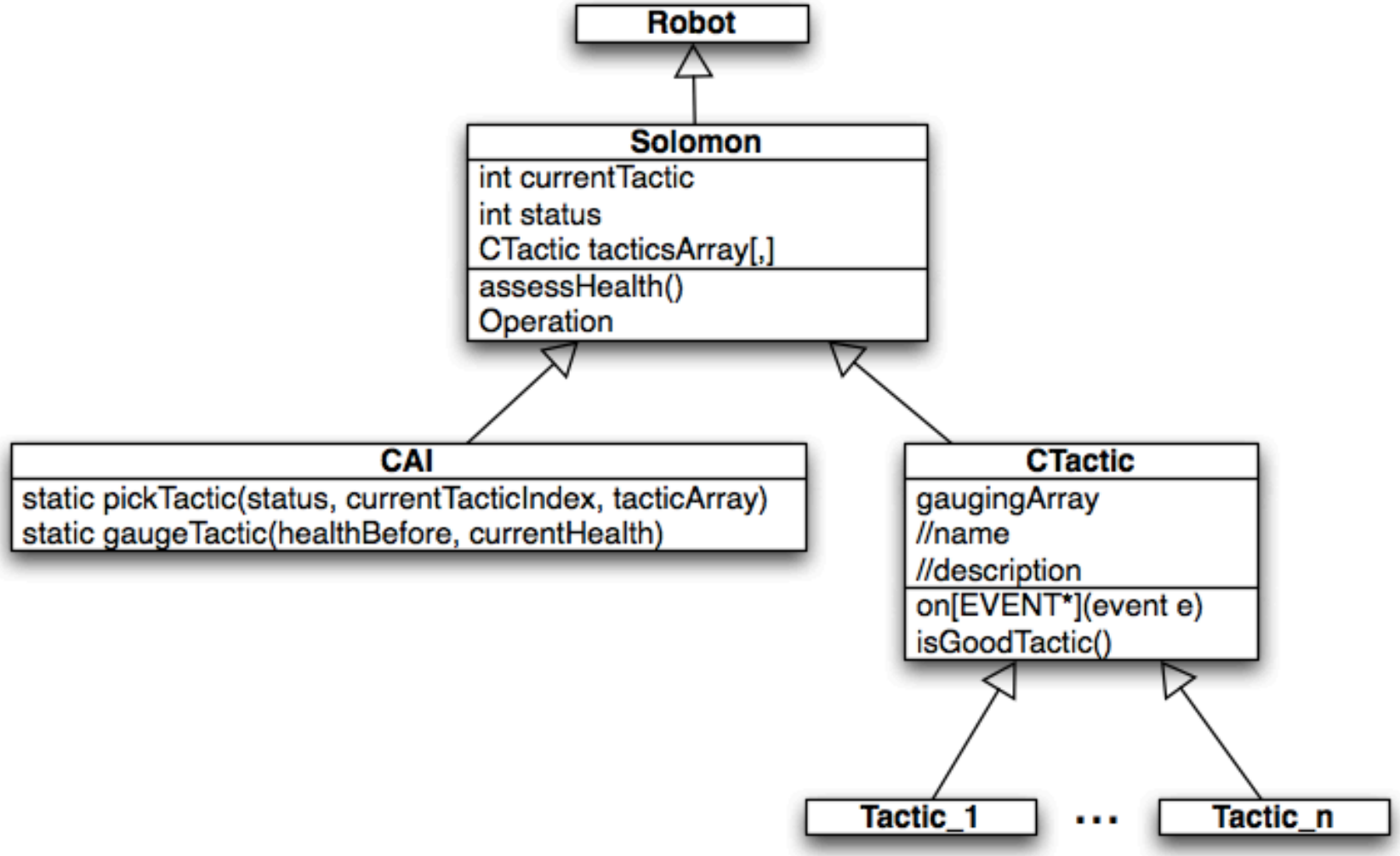
AI Class

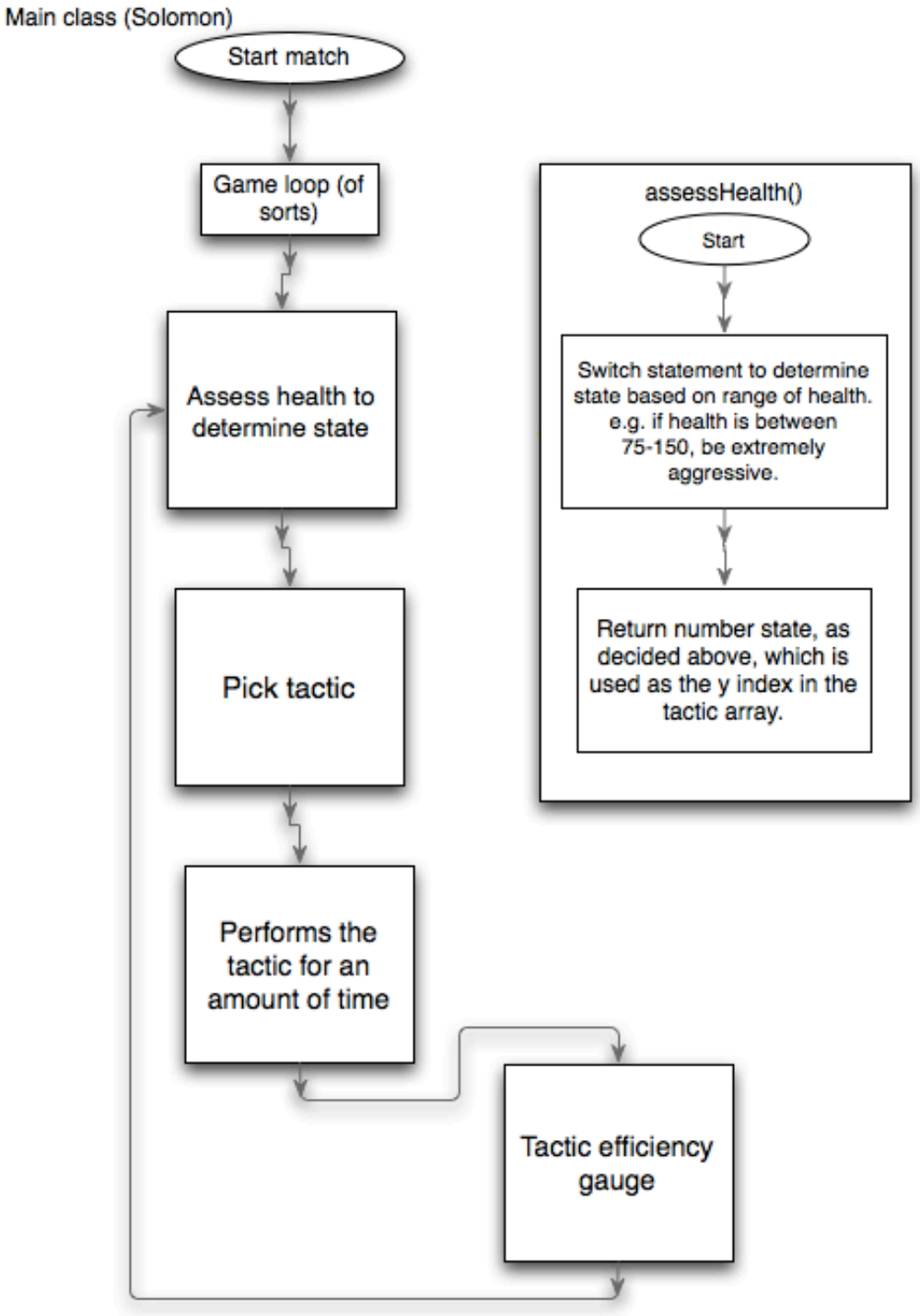
The AI class is Solomon's decision-making component.

//TODO: Finish this.

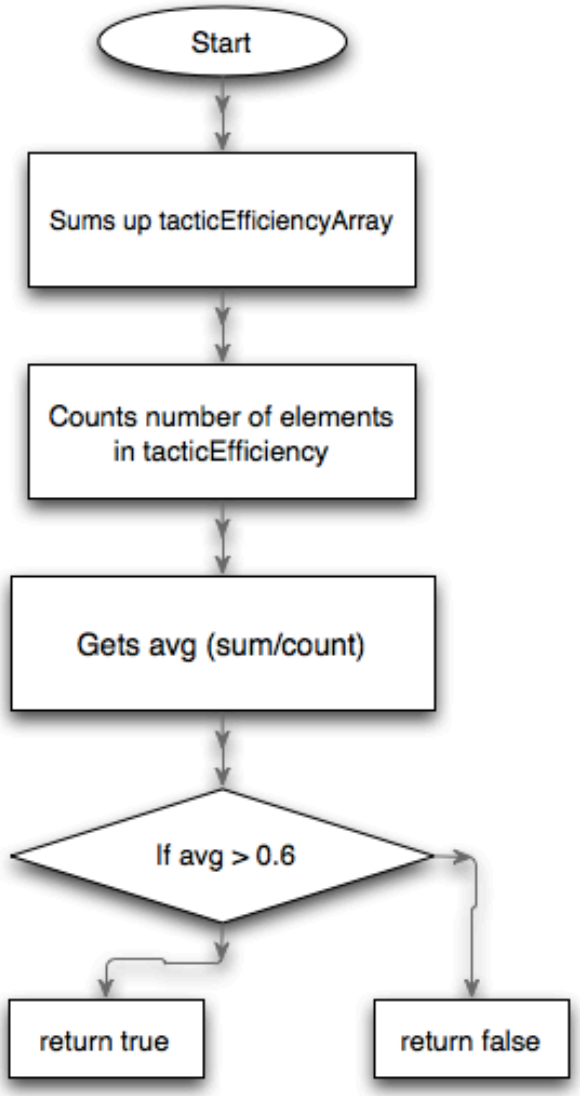
Tactics

CTactic Table	0	1	2
State 1 Extremely Aggressive	Rams the opponent, then moves as far away as possible, firing as it retreats. Then repeats. A tutorial used for non-iterative linear targeting. http://bit.ly/bw2s2r	//TODO	//TODO
State 2 Aggressive	//TODO	//TODO	//TODO
State 3 Defensive	Acts very like wallkiller, that is, Solomon follows the walls and fires inwards. Scans in the corners. Same targeting as ea0.	//TODO	//TODO
State 4 Extremely Defensive	Longest-distance random movement. That is, it consistently moves to a random spot where it is further from the main opponent (within reason).	//TODO	//TODO




Main class (Solomon)
Start match
Game loop (of sorts)
Assess health to determine state
Pick tactic
Performs the tactic for an amount of time
Tactic efficiency gauge
assessHealth()
Start
Switch statement to determine state based on range of health. e.g. if health is between 75-150, be extremely aggressive.
Return number state, as decided above, which is used as the y index in the tactic array.

isGoodTactic()
in Tactic class



pickTactic(status, currentTacticIndex, tacticArray)