The Perfect Swarm

Problem Statement

Friday March 2, 2012
Brian Goodacre, James Jacob, and Bill Rossi
Website: http://tinyurl.com/theperfectswarm or http://sg.sg/yZbcHV

Table of Contents

Table of Contents
Overview
Customers
Requirements
System Functions
  Evident
  Hidden
  Frill
System Attributes
  Details
  Constraints
  Must
  Want
Use Cases
  Actors
  List of Use Cases
  Use Case Diagram
  Use Case Specifications
    Move
    Manipulate Environment
    Sense Environment
    Attack Agent
    Transmit/Receive Information
    Start Simulation
Data Dictionary Document
User Interface Map
  Simulation Start
  Simulation Running
Project Schedule
  Use Case Deliverables
  Rationalize Schedule
    Resources
    Expertise
    Risks
    Team Responsibilities

Overview
Throughout the world, there are a vast variety of situations and environments that are inaccessible to humans. These are highly dangerous terrains that would put any humans at great risk, hard to reach locations that are inhospitable to people, and long-term projects that simply take too much time to allow for human interaction. However, as society and technology progresses, robotic systems have begun to take up these tasks in place of their fragile and expensive counterparts. Unfortunately, these systems are often limited in their abilities. The robots are usually controlled by human operators and often work alone. In respect to many real-world problems, a swarm of independent robots would be much more appropriate. These robots would be able to navigate around mines within a battlefield, operate within the unforgiving pressures beneath the ocean or beyond the atmosphere, and repair large structures and systems throughout the course of months or years without a central authority. Designing the intelligence in a computer-simulated environment will make it possible to tailor an intelligence to the specifications of real-world robots, and discovering the capabilities of swarm intelligence will allow for improving the design of the robots of the future.

Customers
The customers for The Perfect Swarm (TPS) are researchers and government/commercial investors. Researchers will want to utilize the TPS system in order to develop and test new swarm algorithms within an adaptable sandbox application. On the other hand, commercial and government investors will want to employ the system to implement real-world robotic systems. Commercial companies will typically use the swarm intelligence system within domestic and factory environments. Government agencies will most likely use the system within battlefield situations. Each type of stakeholder will have a set of certain requirements for the system, which must be fulfilled in order to the TPS to be both useful and profitable.

The researchers and commercial investors will have the following requirements:
- Simulation infrastructure should be stable across all testing environments.
- System should be able to record statistics about multiple simulations.
- Agents should deploy different swarm intelligence algorithms.
- Agents should choose best swarm intelligence algorithm in a given current scenario.
- GUI should clearly show the current state of any simulation.

The researchers will have the following additional requirements:
- Users should be able to develop and implement new swarm algorithms for testing

The commercial investors will have the following additional requirements:
- Simulation infrastructure should closely resemble a real-world environment.
- Agents should be scalable to receive take real world input.
- Agents should be scalable to give real mechanical output.

Requirements
1. System capable of simulating varied environments
   a. Environments will be composed of many different elements, each with it’s own properties and characteristics
2. System capable of deploying variable number of software agents
3. System capable of collecting metrics on agent algorithm and problem resolution
a. Metrics include: run time, number of agents deployed, number of agents destroyed, areas explored, etc

4. System capable of taking user input
   a. User input to specify problem, algorithm type, and number of agents
   b. User input to modify the environment in order to assist/impede agents

5. System capable of implementing different swarm algorithms based on user input/problem type

6. System capable of visualizing the environment and movement of agents throughout the course of the simulation

7. System capable of having agents move and interact with the environment.

System Functions

Evident
- System will load in new environments from an external text file
- System will prompt user to specify problem, algorithm, and number of agents to be deployed, size of environment
- System GUI will display the interactions between agents and the environment
- System will display data about the algorithm currently running as well as the agents currently active
- System will introduce hostiles into environment in order to hinder agent progress

Hidden
- System will keep track of all agents, both active and inactive
- Agents will communicate amongst each other
- Agents will sense nearby environment
- System will collect data about the algorithm currently running and the agents currently active
- System environment will be capable of influencing agent movement and agent behavior
- System will record agent movements in order to assist study of algorithm efficiency

Frill
- System will implement set of auditory cues in order to represent certain actions taken by agents and the user

System Attributes

Details
- Different environment types will be represented by different colors
- System environment will be simulated by a 2-D array of cells
  - Each cell will be a separate traverse-able area, with it's own environment
- Agent location will be recorded within linked list
- Agents will load different algorithms during execution
- System driver queues agent moves
- System driver clocks the simulation
Constraints
- Agents will not be able to move beyond the environment area specified by the system
- Agents will only be able to communicate with other agents within a certain defined radius
- Only certain environment types will be traverse-able by agents
- Only a certain number of agents will be allowed within each environment cell
- Text file environment will be loaded from must be formatted in specific manner
- Text file agents will be loaded from must be formatted in specific manner

Must
- System has agents which work together to accomplish tasks
- Agents are capable of utilizing different search algorithms
- System has an environment which can be traversed by agents
- System has a GUI which visualizes the actions in play within the environment and allows users to manipulate the environment set-up

Want
- GUI uses images to represent different environments instead of colors
- More than only paving roads, such as building bridges and drilling tunnels
- Agents will be tasked with moving high-value resources to certain locations
- Multiple types of statistics will be kept for executions

Use Cases

Actors
- Agents
- Hostiles
- Environment
- Input Terminal

List of Use Cases
(The numbers correlate to the use cases, i.e. 1. == UC 1)
- Move
- Manipulate Environment
- Sense Environment
- Attack Agent
- Transmit/Receive Information
- Start Simulation
**Use Case Diagram**

**Use Case Specifications**

**Move**

- **Header**
  - Use Case Name
    - Move
  - Use Case Number
    - UC-1
  - Description
    - Utilized by Agent actors in order to traverse the Environment actor domain
  - Purpose
    - Allows Agents to communicate with Environment in order to update Agent movement throughout Environment cells
      - Necessary to simulate Agent movement within the given terrain
  - Requirements Trace
    - 5, 6, 7
  - Subject Area
    - Swarm Intelligence
o Related Use Cases
  ■ Sense_Environment
o Primary and Secondary Actors
  ■ Primary Actor: Agent
  ■ Secondary Actor: Environment
o Technical Requirements
  ■ Need to update Agent_Location array with new cell location
  ■ Need to update Environment_Map array with Agent location
    1. GUI updates with corresponding visuals in appropriate cells
The Perfect Swarm - Goodacre, Jacob, Rossi

- **Body**
  - **Pre-condition**
    - Adjacent Environment cells must be open to facilitate Agent
      1. Environment must not be terrain inaccessible to Agent
    - Agent must be active
  - **Main course**
    1. Agent sends Environment new Agent location coordinates (xValue, yValue)
    2. Environment checks new location coordinates for availability
    3. Environment updates Agent location within Agent_Location array
    4. Environment updates GUI visuals with Agent location changes
      a. Clears old cell location of Agent visual
      b. Updates new cell location with Agent visual
    5. Environment sends Agent confirmation of location change
    6. Agent update current location (xValue, yValue)
  - **Alternate courses**
    - Agent moves into a now occupied cell
      1. System sends signal to both Agents that they crashed and are now broken.
      2. Environment updates with crash site.
    - Environment cell unavailable
      1. Agent sends Environment new Agent location coordinates (xValue, yValue)
      2. Environment check new location coordinates for availability
      3. Environment send Agent message that cell location unavailable

- **Trailer**
  - **Issues**
    - How to visualize multiple Agents within a particular cell in GUI
    - What is the maximum amount of Agent allows within a particular Environment cell
  - **Assumptions**
    - Agents only capable of movement into adjacent cells
      1. Can only move one cell at a time
    - Agents are incapable of moving off the Environment map
  - **Design comments**
    - Each agent will be implemented as a thread to help simulate individual memory
    - Moves will be queued by the system driver. In other words, they occur synchronously.
  - **Change Log**
    - n/a

**Manipulate Environment**
- **Header**
  - **Use Case Name**
- Manipulate_ENVIRONMENT
  - Use Case Number
    - UC-2
  - Description
    - Utilized by Agent actors to update Environment actors cells that they are currently interacting with
  - Purpose
    - Allows Agent to communicate with Environment in order to facilitate Agent activities that may affect the Environment
      1. Necessary in order to facilitate Agent’s paving roads, moving resources, and building structures within the Environment
  - Requirements Trace
    - 1, 2, 5, 6
  - Subject Area
    - Swarm Intelligence
  - Related Use Cases
    - Sense_ENVIRONMENT
  - Primary and Secondary Actors
    - Primary Actor: Agent
    - Secondary Actor: Environment
  - Technical Requirements
    - Need to update Environment_Map array with new details depending on Agent actions
      1. GUI updates with corresponding visuals in appropriate cells
The Perfect Swarm - Goodacre, Jacob, Rossi

- **Body**
  - Pre-condition
    - Agent must be active
    - Environment cells must contain elements capable of being interacted with
  - Main course
    1. Agent send Environment new Environment cell characteristics (Cell color)
    2. Environment updates GUI visuals with new cell characteristics
      a. Updates all cells influenced by Agent activities with appropriate visuals
    3. Environment sends Agent confirmation of cell characteristic change
  - Alternate courses
    - None
- **Trailer**
  - Issues
    - How should Environment cell characteristics be represented within GUI
  - Assumptions
    - Agent only capable of completing one interaction per clock cycle
  - Design comments
    - The agent can only find information about the environment through this use case so the system driver can maintain control over all the agents and what they can do.
  - Change Log
    - n/a

**Sense Environment**

- **Header**
  - Use Case Name
    - Sense_Environment
  - Use Case Number
    - UC-3
  - Description
    - Utilized by Agent actors in order to observe the local Environment terrain
  - Purpose
    - Allows Agents to communicate with Environment in order to update the Agent’s internal map of the Environment
      - Necessary to allow Agents to choose the best possible actions in order to achieve the overarching goal
  - Requirements Trace
    - 1, 2, 5, 6
  - Subject Area
    - Swarm Intelligence
  - Related Use Cases
    - Manipulate_Environment
  - Primary and Secondary Actors
    - Primary Actor: Agent
    - Secondary Actor: Environment
The Perfect Swarm - Goodacre, Jacob, Rossi

- Technical Requirements
  - Need to update Agent’s LocalCell_Map array with current terrain

- Body
  - Pre-condition
    - Current local terrain must be in conflict with Agent’s internal map
  - Main course
    1. Agent sends Environment query for local Environment cell values (xValue, yValue)
    2. Environment sends Agent cell terrain values (terrain[9])
    3. Agent checks if new terrain values in conflict with LocalCell_Map
    4. Agent updates internal LocalCell_Map array with new terrain value
  - Alternate courses
    - No change to local terrain
      1. Agent sends Environment query for local Environment cell values (xValue, yValue)
      2. Environment sends Agent cell terrain values (terrain[9])
      3. Agent checks if new terrain values in conflict with LocalCell_Map
      4. Agent deletes recently sent terrain values

- Trailer
  - Issues
    - How should local Environment terrain be stored within Agent memory
    - How much of past traversed terrain should be stored within Agent memory
  - Assumptions
    - Agent always knows where it is on Environment map
  - Design comments
    - Similar to manipulate environment. Once again, the system
  - Change Log
    - n/a

Attack Agent
- Header
  - Use Case Name
    - Attack_Agent
  - Use Case Number
    - UC-4
  - Description
    - Utilized by Hostile actors in order to hinder Agent actor activities
  - Purpose
    - Allows Hostiles to communicate with Agents in order to either destroy Agent or halt any actions it is currently taking
      - Necessary to allow Hostiles to perform their main function
  - Requirements Trace
    - 1, 2, 5, 7
Subject Area
- Swarm Intelligence

Related Use Cases
- ManipulateEnvironment
- SenseEnvironment
- Transmit/Receive Data

Primary and Secondary Actors
- Primary Actor: Hostile
- Secondary Actor: Agent

Technical Requirements
- Need to check Agent memory in order to realize what action Agent is currently undertaking

Body
- Pre-condition
  - Hostile must be occupying Environment cell adjacent to Environment cell occupied by target Agent
- Main course
  1. Hostile checks Agent memory so as to realize current activity being undertaken
  2. Hostile sets Agent activity variable to inactive
- Alternate courses
  - Hostile not capable of interfering with current Agent activity
    1. Hostile checks Agent memory so as to realize current activity being undertaken
    2. Hostile does not interfere with Agent activity

Trailer
- Issues
  - What kind of hostiles can affect the Agent
  - Are Hostiles capable of moving around the Environment map
  - Are agents capable of interfering with Hostile activities
  - Are Hostiles capable of going inactive
- Assumptions
  - Agents are only affected by Hostiles corresponding to the task they are currently undertaking
  - Hostiles do not need an intelligence system
- Design comments
  - The movement code and logic will be similar to the agent’s code but this hostile attacking will be a different goal for the hostile agents.
- Change Log
  - n/a

Transmit/Receive Information
- Header
  - Use Case Name
Transmit_Receive_Information
- Use Case Number
  - UC-5
- Description
  - Utilized by Agent actors in order to communicate their location and surrounding with other Agent actors
- Purpose
  - Allows Agents to communicate local Environment information to nearby Agents
    - Necessary to allow Agents to build a swarm intelligence pattern
- Requirements Trace
  - 1, 2, 5, 7
- Subject Area
  - Swarm Intelligence
- Related Use Cases
  - Move
  - Sense_Environment
- Primary and Secondary Actors
  - Primary Actor: Agent
  - Secondary Actor: Agent
- Technical Requirements
  - Need to update Agent memory with Environment information from other Agents
    1. Need a 2D array of the entire Environment that can be populated

Body
- Pre-condition
  - Agents must be within a required Environment cell radius of each other
- Main course
  1. Agent 1 checks localCell_Map array within memory for nearby Agents
  2. Agent 1 queries nearby Agent 2 for transfer of information
  3. Agent 1 sends local Environment cell data to Agent 2 (localCell_Map[9])
  4. Agent 2 sends its own local Environment cell data to Agent 1
  5. Agent 1 updates internal memory with local Environment cell data from Agent 2
- Alternate courses
  - No Agents nearby
    1. Agent 1 checks localCell_Map array within memory for nearby Agents
    2. Agent 1 continues with its business
  - No new Environment data sent
    1. Agent 1 checks localCell_Map array within memory for nearby Agents
    2. Agent 1 queries nearby Agent 2 for transfer of information
3. Agent 1 sends local Environment cell data to Agent 2 (localCell_Map[9])
4. Agent 2 sends its own local Environment cell data to Agent 1
5. Agent 1 continues with its business

• Trailer
  o Issues
    ■ Will Agents retain all Environment information given to them or will they only retain the more recently sent data
    ■ How close to Agents need to be to each other in order to communicate
    ■ How will Agents know if they already have the information being sent
  o Assumptions
    ■ Communication occurs through function calls.
  o Design comments
    ■ Communication to allowed agents occurs through asking the system driver who they can communicate too
  o Change Log
    ■ n/a

Start Simulation
• Header
  o Use Case Name
    ■ Start_Simulation
  o Use Case Number
    ■ UC-6
  o Description
    ■ Utilized by Input_Terminal actor to create and begin simulation runs and adjust simulation settings
  o Purpose
    ■ Allows Input_Terminal actor to communicate with Environment and Agent actors in order to organize simulation set-up
    ■ Necessary to allow product users to utilize the Input Terminal and use the product
    ■ Allows Input_Terminal actor to communicate with users in order to show them algorithm and simulation metrics
  o Requirements Trace
    ■ 3, 4, 6
  o Subject Area
    ■ Swarm Intelligence
  o Related Use Cases
    ■ none
  o Primary and Secondary Actors
    ■ Primary Actor: Input_Terminal
    ■ Secondary Actor: Agent
    ■ Secondary Actor: Environment
  o Technical Requirements
    ■ Need to create panel within GUI which will represent Input_Terminal and
take commands from users

- **Body**
  - Pre-condition
    - No simulation is currently running
    - Input configuration has been entered into Input_Terminal
  - Main course
    1. Input_Terminal sends Environment all initial parameters (size, cell_Terrain[], Hostile_presence[]) 
    2. Input_Terminal sends Agents all initial parameters (agentTotal, xLoc[], yLoc[])
    3. Input_Terminal begins simulation with all Agents running in parallel
    4. Input_Terminal display simulation metrics as simulation progresses
  - Alternate courses
    - none

- **Trailer**
  - Issues
    - How will Input_Terminal display simulation metrics
  - Assumptions
    - User always enters simulation settings properly
  - Design comments
    - initiated by the user
  - Change Log
    - n/a

**Data Dictionary Document**

**Alphabetical order**

- **Agent**
  - Each thread with its own code and memory space will represent a robot in the simulation. These threads are called agents.

- **Algorithm**
  - A set of rules agents will follow when making decisions on their next move.

- **Crash Site**
  - A terrain type that is impassable because two agents collided at that spot.

- **Fire fights**
  - A terrain type that if an agent enters, the agent becomes broken and cannot send communication any more.

- **GUI**
  - The visual representation of the current environment during the simulation.

- **Hostiles**
  - Threads in the simulations that try to find agents and break them.

- **Mountains**
  - A terrain type that cannot be navigated by agents and is not part of the path from starting point to ending point.
• Moves
  o Agents can choose to advance one grid position in two dimensions each round.
• Paved Land
  o A terrain type that can be navigated by agents and is part of the path from starting point to ending point.
• Round
  o All agents queue their moves and the moves all get executed at the same time based on the clock of the system driver.
• Simulation
  o One instance of agents moving through the environment based on initial conditions set by the user
• System Driver
  o The clocked class that queues moves from the agents and determines which ones executes. It then tells the agents and the environment the results of the moves.
• Unpaved land
  o A terrain type that can be navigated by agents and is not part of the path from starting point to ending point.
• Water
  o A terrain type that cannot be navigated by agents and is not part of the path from starting point to ending point.

**User Interface Map**
The Perfect Swarm is mainly a simulation so user input will be at a minimum. The interfaces the user is concerned with is the initial simulation start screen and the ongoing simulation screen. The output results will be given in a file so there is no interface for that.
Simulation Start

The Perfect Storm

- # of Agents
- Cell Count
- Terrain File Browse
- Start Simulation

Simulation Specifications Screen UC00001

Start Simulation

Simulation Running Screen UC00002

- Play
- Restart
- Pause
- New Simulation
**Simulation Running**

![Simulation Interface](image)

**Project Schedule**

**Use Case Deliverables**

<table>
<thead>
<tr>
<th>#</th>
<th>Use Case</th>
<th>Deliverable 1-3/30</th>
<th>Deliverable 2-4/13</th>
<th>Deliverable 1-5/04</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC-1</td>
<td>Move</td>
<td>Agents can transverse environment from beginning to end</td>
<td>Agents work together to traverse and explore</td>
<td>Agents learn about environment and avoid danger</td>
</tr>
<tr>
<td>UC-2</td>
<td>Manipulate</td>
<td>X</td>
<td>Agents can pave land</td>
<td>Agents</td>
</tr>
<tr>
<td></td>
<td>Environment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC-3</td>
<td>Sense Environment</td>
<td>System can tell agents its surroundings.</td>
<td>System takes into account environment (i.e. no information can be sent through mountains)</td>
<td>X</td>
</tr>
</tbody>
</table>
The Perfect Swarm - Goodacre, Jacob, Rossi

<table>
<thead>
<tr>
<th>UC-4</th>
<th>Attack Agent</th>
<th>X</th>
<th>Have hostiles in environment</th>
<th>Implement algorithms for hostiles to find agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC-5</td>
<td>Transmit/Receive Information</td>
<td>Agents can share their knowledge</td>
<td>Agents combine most recent information</td>
<td>Agents can send information to all agents by going through local agents</td>
</tr>
<tr>
<td>UC-6</td>
<td>Start Simulation</td>
<td>System creates/load environment and agents. GUI shows environment.</td>
<td>Customized settings</td>
<td>Allow stopping of simulation and manually changing of environment</td>
</tr>
</tbody>
</table>

**Rationalize Schedule**

**Resources**
The team is using a SVN repo to allow easy updates of code. All three team members are using the Eclipse IDE and are programming in Java.

**Expertise**
All team members have experience programming in Java. James has additional knowledge with GUIs in Java and Brian has knowledge of programming synchronous system drivers. All team members will be learning and experimenting with swarm intelligence algorithms initially in the first two iterations and then implementing their own swarm intelligence algorithms for the final iteration.

**Risks**
The team does not have previous experience with artificial intelligence so leaving enough time to experiment with AI is very important.

**Team Responsibilities**
Brian, the leader, is responsible for pushing for deadlines and adjusting the goals and deadlines appropriately depending on progress. He is also responsible for making sure communication is good amongst all three team members.

<table>
<thead>
<tr>
<th>#</th>
<th>Use Case</th>
<th>Brian Goodacre (leader)</th>
<th>James Jacob</th>
<th>Bill Rossi</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC-1</td>
<td>Move</td>
<td>Record moves in the environment</td>
<td>Algorithms to have agents move</td>
<td>Algorithms to have agents move</td>
</tr>
<tr>
<td>UC-2</td>
<td>Manipulate Environment</td>
<td>Record manipulations</td>
<td>Advanced algorithms</td>
<td>Basic algorithms</td>
</tr>
<tr>
<td>UC-3</td>
<td>Sense Environment</td>
<td>Provide agents information of their surroundings</td>
<td>Advanced algorithms</td>
<td>Basic algorithms and store information</td>
</tr>
<tr>
<td></td>
<td>Attack Agent</td>
<td>Record attack</td>
<td>Algorithms for agents to avoid</td>
<td>Algorithms for hostiles to attack</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------</td>
<td>--------------------------------------------</td>
<td>-------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>UC-4</td>
<td>Attack Agent</td>
<td>Record attack</td>
<td>Algorithms for agents to avoid</td>
<td>Algorithms for hostiles to attack</td>
</tr>
<tr>
<td>UC-5</td>
<td>Transmit/ Receive Information</td>
<td>Provide system that allows for communication</td>
<td>X</td>
<td>Make agents send information and receive it to allowed agents</td>
</tr>
<tr>
<td>UC-6</td>
<td>Start Simulation</td>
<td>Create environment and agents</td>
<td>Allow through GUI</td>
<td>X</td>
</tr>
</tbody>
</table>