Agent Race - Final Report

Andrea Sperandio
andrea.sperandio4@studio.unibo.it

January 2022

Agent Race is a multi-agent system designed to deepen its engineer's skills on various topics of the Autonomous Systems course, while providing entertainment to its human users.
This MAS represents a society where its agents' purpose is to win a running race performed in straight lanes. When the system is deployed, many agent-athletes are spawn with different characteristics and each one of them will try its best to exploit its strengths and the mutable environmental conditions to arrive first to the finish line.
Each agent perceives itself as well as the surrounding environment and the other competitors in the race and decides the best move to play next amongst many options, like sprinting, slowing down, stopping at a refreshment point when available.
The environment plays a key role in the competition's outcome influencing the behaviour of the agents through its traits and artifacts, like race length, weather conditions, agent's lane conditions, refreshment points availability.

1 Goal/Requirements

The project's goal is to design and build a system where agents compete to win a race, using and exploring some of the technologies introduced by the Autonomous Systems course.
The idea is inspired by the real world running races, where competitors arrive to the running track, meet with the organizer to register for the race, get ready at the starting position and compete to win. Some real life aspects are replicated in this system, others are simplified: for example, while it stays true that some runners may arrive late and therefore being excluded from the race, for simplicity competitors run in straight lanes with different characteristics.

To design and build such a system it is important to have some race management and running experience and to study which the key factors that may influence the runners' behaviour and the overall race outcome are.

Since the user entertainment is a core requirement for this project, it is crucial to provide it with a graphic interface through which it is possible to visually follow the progress of the race as well as to interfere with its course. This way the human user can become an active part of the system as well, cheering for some runner and modifying the environment to help and favour him.

1.1 Scenarios

In order to explain how a user can interact with the system, it is best to provide a working example first.

Once the organizer holds a competition, athletes start coming to him for the registration procedure. The number of participants is limited by the organizer's rules, so it is possible that not every runner will be allowed to participate. Indeed, another possibility is that no runner shows up and the event is cancelled! When a runner completes the registration procedure, he goes to the assigned lane and gets ready waiting in the starting position. Filled the available lanes or reached the starting time, the organizer starts the race and each runner moves along the track trying to win. Just like in the real world, every runner is different from the others and this aspect is modelled by various traits like stamina, fitness shape, favourite weather conditions and predisposition to injury. The race track is designed to somehow reflect the real one too, so it might have faults in the path and offer refreshment points to the runners. Finally, the weather conditions influence the runners' mood and visibility, sometimes deceiving them from an easy victory by hiding a close and more tired opponent. In this unpredictable situation, every runner tries his best to beat the others, sometimes having to decide between multiple available strategies, like stopping at a refreshment point, which causes him to waste time but also to recover for a better race continuation and ending. Runners can complete the race or withdraw due to an injury. At the end of the race it is the organizer that proclaims the winner and checks the arrivals list, while keeping track of each runner's ending time.

The human user can follow the race progress through an interactive graphic interface, which shows the race track, the mutable weather conditions, the presence of faults or refreshment points along each race lane and, of course, the location of the runners on the track. The user can just watch the competition, thus becoming a spectator, or interfere with it. In particular, he can change the weather conditions, affecting the runners' visibility and mood, and modify the track lanes, adding or removing faults and refreshment points. Before launching the system, it is possible to set a parameter (one for the organizer and another one for the runners) that controls the agents' output verbosity level, making it possible for the user to inspect the feelings and decisions of each single runner or to exclude this information from the console log.

1.2 Self-assessment policy

- The quality of this software should be evaluated according to its organization: key aspects are separation of concepts, reusability and understandability. During the assessment procedure one should keep in mind questions like: are the technologies

chosen suited for a MAS design and development? are the core concepts well separated and encapsulated? did the developer use specific design patterns to solve common problems?

- Given the initial goals and requirements, the effectiveness of this project outcomes should be assessed on the fulfilment of those goals. One should validate aspects like: is this a MAS? are its agents autonomous entities? does the system somehow emulate a real running race? is the human user part of the system?

2 Requirements Analysis

This project aims at the creation of a semi-closed system, meaning that only a human user can interact with it through the graphic interface, but it is not supposed to interact with other software modules or systems. So, the main requirements of this system are those of a self organizing closed system with the addition of observability and responsiveness to the external user's actions.

In order to model such a system one can choose amongst many different paradigms, the one that best suites the requirements though is the multi-agent system.

The chosen technology for this project is JaCa (Jason + CArtAgO), which allows the designer and developer to model the organizer and the runners as agents and the environment elements (like the race track, the refreshment points, the chronometer) as artifacts. There is still some abstraction gap between the scope of the project and the technologies used: for example the weather is an aspect of the environment, more than of one of its artifact (like the race track).

3 Design

In the design phase it is clear that the project's goals require the modelling of different entities that interact with each other. Now an analysis of several logical aspects is presented.

The main entities required for this project are:

- Organizer
  The Organizer of the race is the one that takes care of each aspect of the event and keeps everything together. This is the entity that creates the race competition, opens/closes the registration phase, starts the race and draws up the arrivals. Its main features are related to communication and coordination.

- Runner
  Runners are those that compete to win the race. This entity is decorated with multiple characteristics which influence the way it reasons and takes decisions. Its main features are bounded to logical reasoning.

- Environment
  The Environment is the layer that allows for communication and actual racing. Logically, it is the place where the Organizer and the Runners meet and where the race takes place. It provides the communication field, a race track, a chronometer, mutable weather conditions... Its main features concern communication and localization.

3.1 Structure

The Organizer entity is the one that creates the race event, so it has to decide the number of available lanes (i.e. the maximum number of runners allowed) and the length of the race. Other than that, it takes care of the different phases of the event, so first of all it has to open and close the registration phase and decide if and when to start the competition. This is also the entity that informs the system's human user of which race stage is currently being processed. Finally, it keeps track of withdrawn and arrived runners, of the arrival time (through a chronometer) and draws up the arrivals.

The Runner entity needs to locate the track and register with the organizer first. It is endowed with many personal traits and different behaviours. His main characteristics are now presented:

- injury probability: each runner is different and this aspect models the risk of getting injured during the competition. An injured runner has to withdraw from the event. This probability remains stable during the race, but can become milder if the runner stops at a refreshment point;

- stamina: it represents the resistance and remaining energies of the athlete. During the race the runner's stamina level decreases and the only way to gain it back is by stopping at refreshment points or getting lucky with weather conditions. This last aspect will be explained in a while;

- fitness: regardless of his stamina level, each runner might be or not be particularly in health. This trait is modelled by the fitness indicator and never changes during the event. It is an aspect related to how trained for a competition and healthy the runner is;

- favourite weather: just like humans, this software runners have preferences over the weather conditions. This might reflect on their stamina level, increasing it when the external weather conditions meet the one the runner prefers. Of course, this aspect too never changes during the race.

The Environment plays a key role for the entities communication. Moreover, it offers the runners a track where to race in and other exploitable artefacts. The race track consists of multiple parallel straight lanes which can host one runner each, every lane is different from the others. A lane is made up of a determined number of steps (which represents the race length) and a runner must go through each one of them sequentially

in order to complete the race. So, the runner's speed depends on the time he takes to move from one step to the next one. Every step is characterized by a status:

- valid: it is a normal step and stays neutral to the runner;
- faulty: this step can be dangerous for the athlete, causing him injuries. The injury probability is influenced by the runner's predisposition to it, but also by its speed and by the weather conditions, which might make the track slippery or barely visible;
- refreshment point: the step offers a chance for recovering to the athlete, which might decide to stop (and use some extra time) to regain some stamina and protection against injuries.

Finally the Environment provides the organizer with a Chronometer, which will be used to take the athletes' running time, and the overall event with variable weather conditions. These influence the runners' visibility and risk of injury over faulty steps. In particular, the visibility is used to detect faults in the lanes, refreshment points and the other runners.

This design model has been chosen inspired by the real world running events and because it represents one of the possible solutions that better suits the project goals. Furthermore, the separation of concepts here used easily allows for future modifications, improvements and additions. This way it is easy to isolate a system part and re-use it in other projects or simply re-model it without producing an heavy impact on the whole system. Nevertheless, the project aims at building a MAS society, so it is obvious and intrinsic that every system component interacts with and has an impact on the others, directly or indirectly.

3.2 Behaviour

During the design, each entity has been endowed with internal goals so that the resulting system behaviour is in line with the project goals. These internal goals are not shared between entities (i.e. the organizer ignores the runners' goals and vice versa), but lead to interactions and responses that produce the wanted result in a cooperative (between organizer and runners) and competitive (between runners) way.

The organizer's specific goals are to create and manage the race event in all of its stages. So, his behaviour can be summarized by the following activity diagram:

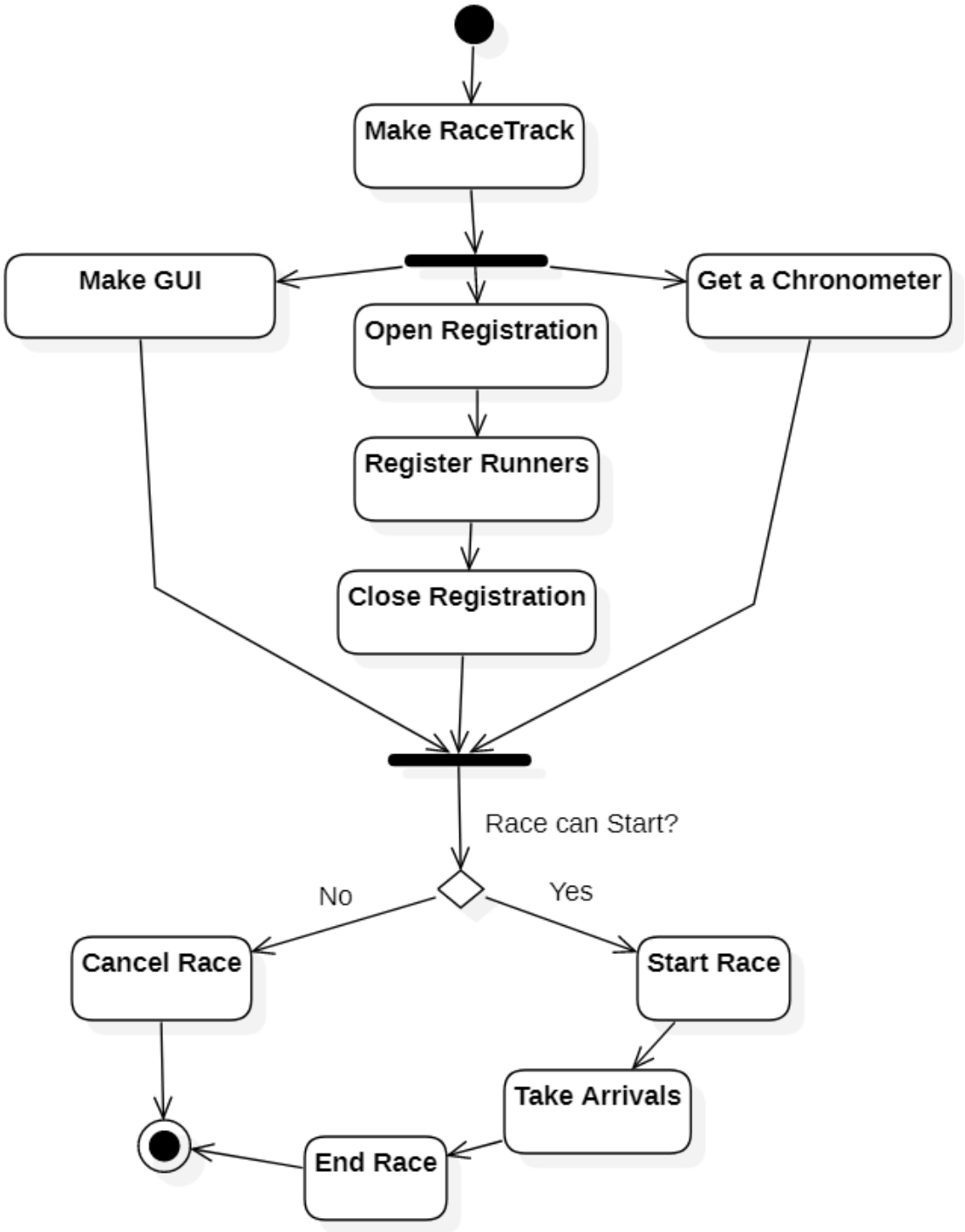

Figure 1: Organizer Activity Diagram

The organizer takes care of the event initialization by making the race track, finding a chronometer and providing a GUI to the human user. Meanwhile, he lets the runners register for the event and decides whether to start the race or not. After that, his behaviour is focused on following the competition and drawing up the arrivals.

The runners' specific goals are to attend the event and try to win the competition. So the following activity diagram shows their main jobs:

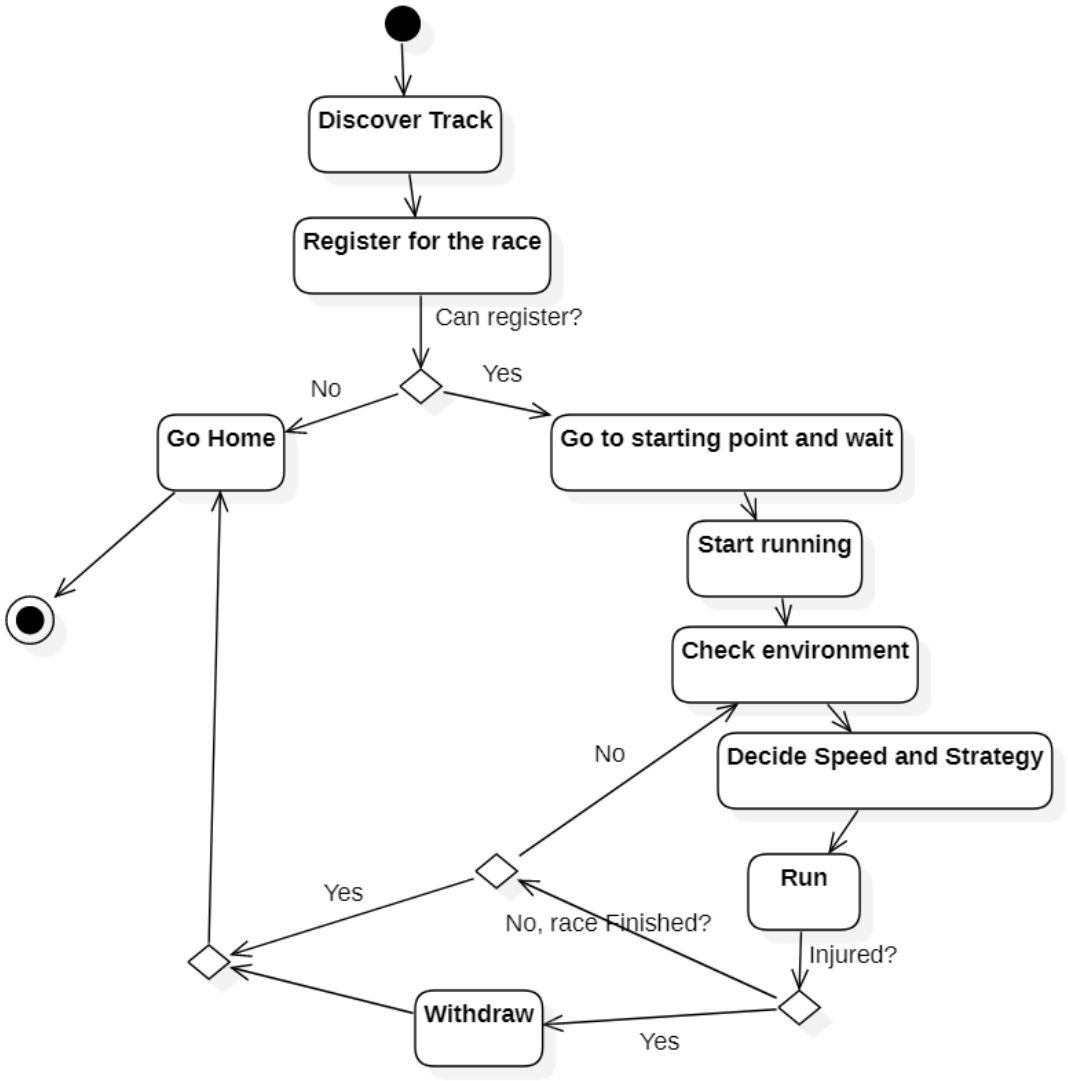

Figure 2: Runner Activity Diagram

As shown in figure 2, first of all the runner shows up for the race and tries to register. Then, he reaches the starting position and waits for the organizer to actually start the

race. At that point, he constantly evaluates the environmental conditions, as well as the other runners' position and makes decision regarding his strategy and whether to stop (at a refreshment point, when available), increase/decrease his speed or sprint. For each step, if he doesn't get injured, he repeats the same pattern until the end of the race. During the design, it is important to reason about in which ways the runner can perceive the environment and what the parameters he uses to make decisions are. Therefore, here are now discussed the main reasonings he can make:

- stop at a refreshment point: in order to decide whether to stop or not at a refreshment point, the runner first evaluates if he's not close to the end of the race and if he feels low on stamina or his risk to get injured is high. If one these conditions is met, he then decides to stop if he feels very tired (too tired to run), or he's close to another runner and the race is still long to finish (so he can recover his strength and beat him in the long run) or he's not in the first three positions (so he's already virtually out of the podium). In all of the other cases, he thinks that the best move for him is to skip the refreshment point and move on;

- slow down: the runner thinks it is best to slow down when he's exhausted or he's just tired and he can't see any other competitor close to him. The other situation where he might want to decrease his speed is when he sees a faulty lane step. In that case, he takes into consideration his injury probability, the remaining race length and his will to risk an injury in order to get on the podium: if he is amongst the first three runners and sees a runner close to him, he risks it and doesn't decelerate;

- increase speed: when a runner feels high on his stamina level he thinks about increasing his speed. That happens when he's close to the end of the race or he doesn't feel he'll easily get injured or he doesn't see a faulty step ahead or, again, when he's in good position and sees other runners close to him;

- sprint: a runner sprints only when the race is almost over and he sees another runner next to him or he can see a refreshment point ahead and feels in good shape and energized enough to sprint.

On the other end, the environment doesn't have any specific goal, but it is designed to react to the other entities interactions and has its own behaviour. The interactions between entities will be discussed in the next section. The environment status might be altered by the human user's actions on the GUI or can change dynamically by itself. The word environment here is being used to represent the space where entities are located, interact and move, as well as its conditions (weather and faults in the track lanes) and its manufacts, like the chronometer and the refreshment points. These traits are now presented individually:

- space: this aspect is modelled through the race track. This represents the space where the runners move and offers them personalized lanes. Faults and refreshment points show their behaviour only when runners interact with them, possibly causing injuries or offering stamina gain and injury protection respectively;

- weather: the weather may change randomly during the race and determines the runners' visibility;
- chronometer: the chronometer is started by the organizer and is used to take the runners' arrival time;
- GUI: the GUI is created by the organizer and may be used by the human user to alter the fate of the race. Its behaviour is to respond to the user interactions altering the environmental conditions, i.e. the weather or the lane steps' status.

Since the organizer and the runners entity can be thought of as agents (autonomous pro-active entities that encapsulate control and operate inside the environment), and the environment hosts entities that can be represented as artifacts (passive components built and used by the agents to support their activities), the overall design choice suggests to adopt the A&A model.

3.3 Interaction

Since 3 is the number of different logical entities identified, it is now discussed all of the main interactions between every pair of entities and then it is presented a simplified sequence diagram for the race event.

- Organizer-Environment Interaction

  The Organizer creates the artifacts of the Environment, like the race track, the race GUI and the Chronometer and links the GUI with the track. Once everything is set up, during the registration phase the organizer asks the track to assign the registering runner to the first available lane. Then, he starts both the race and the Chronometer and when runners arrive to the end, the track informs the organizer of the arrival. At that point he can confirm the arrival and take the runner's finish time thanks to the Chronometer.

  In the A&A model the event the track signals to the organizer when a runner completes the race corresponds to the organizer's percept over the race.

- Organizer-Runner Interaction

  These two entities communicate mainly during the registration phase. When the organizer opens it, he informs all the runners and, those who can, present themselves to get registered. The registration request can complete successfully or fail due to the already closed registration phase, i.e. the runner arrived late or every available lane has been already assigned. Once the registration phase is closed, the organizer starts the race and again informs all the runners: this corresponds to the starting shot. During the competition, some runners may get injured and therefore inform the organizer of their withdrawal. Finally, the organizer tells the runners when the race is finished, so that they can go back home.

- Runner-Environment Interaction

  Initially, the runner looks for the track and, once registered, moves to the starting point. After the race start, he studies the environment looking for the presence of other runners close to him and faults and refreshment points along his track lane. When he encounters a fault he may get injured, while when he stops at a refreshment point he regains stamina and becomes less prone to injuries. The weather has an impact over his mood and visible field, while the other runners' presence influcence the way he'll make decisions.

  In the A&A model the data the track signals to the runner like position, presence of close runners, faults, refreshment points and weather conditions correspond to the runner's percepts over the race. And again, the actions the runner does, like moving or recovering at a refreshment point, are modelled as communication events between entities.

Finally, there are many intra-environment interactions, i.e. between its components. For example, everything that happens on the track is then reflected over the GUI and vice versa, so that every human user's action has an impact both over the model of the system and its GUI, in a MVC perspective.

The following Sequence Diagram (figure 3) illustrates a simplified interaction between the organizer, a runner and the race track. First of all, the organizer creates the track and then informs the runner of the opening of the registration phase. When the runner registers, the organizer commumnicates with the track in order to assign him a race lane and then gives the runner a reply. After the organizer closes the registration phase, he decides to start the race and informs the runner, which starts moving through the track while sensing the environment. When the runner finishes his race, the organizer assigns him an arrival position. Finally, the organizer informs the runner that the race event is finished and disposes the race track.

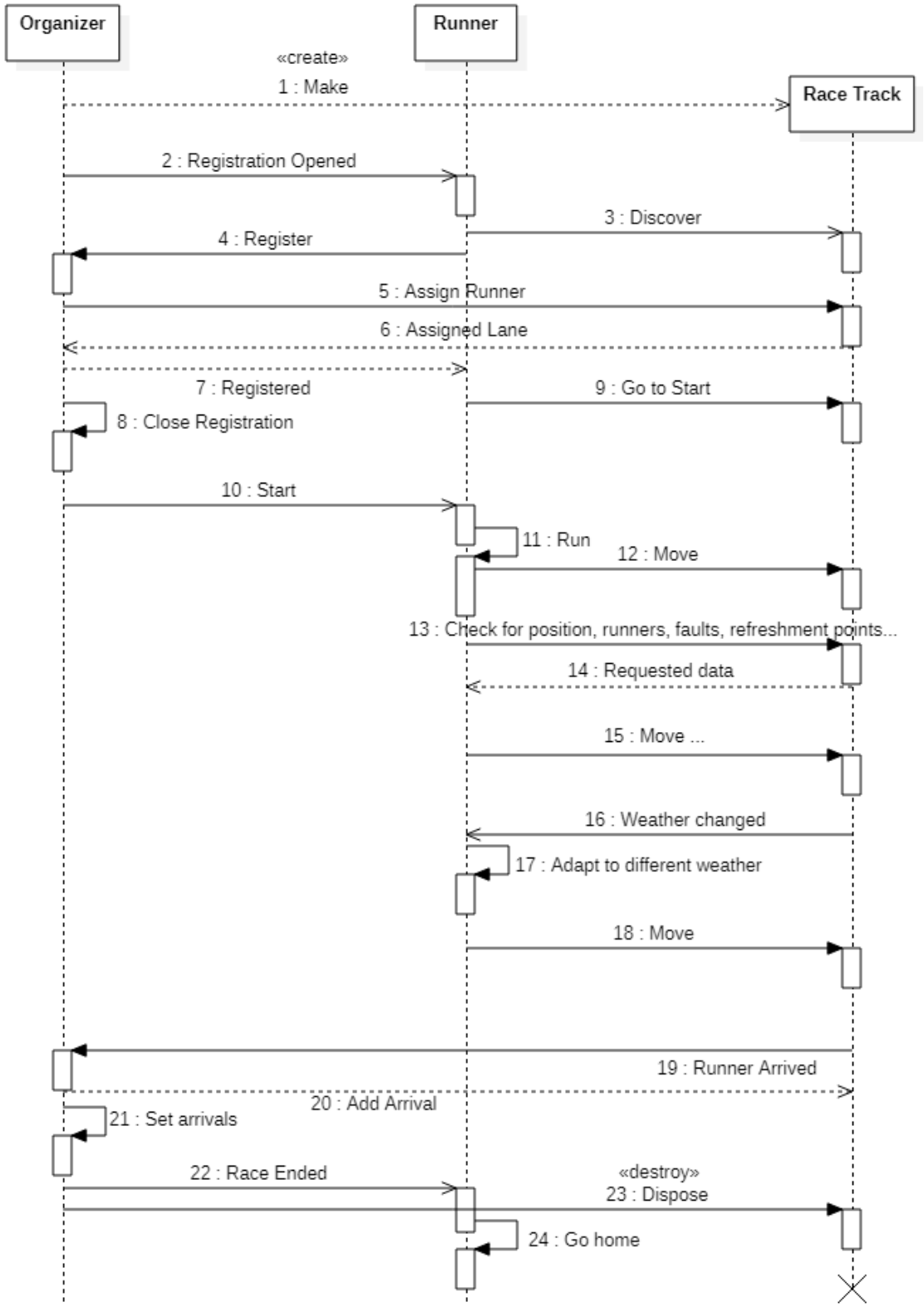

Figure 3: Race Sequence Diagram

4 Implementation Details

During the implementation, in addition to the course slides and the online programming documentation, these articles were particularly helpful: [3], [2], [1].

The produced software is supplied with Javadoc documentation and follows the JaCa model, so it is easy to understand and extend.

A relevant implementation choice was made with reference to the coupling of the race track and the race GUI. In fact these two artifacts are not double linked, but the link is one way from the GUI to the track. This solution was adopted because it highlights that the race event doesn't need a GUI to properly work, but it is helpful to he human user only (to follow the race and interact). Therefore, everything that happens on the GUI side by the human actions is then transmitted to the track and there managed. On the other end, it is the GUI that monitors changes in the race via an Observer pattern and displays every update. This is not the only possible solution of course, but this way the GUI can be thought of as a sort of plug-in for the system, which expands it with more functionalities.

Another implementation detail worth explaining is the management of the runner's visibility. As already stated before, it is influenced by the weather conditions. Its behaviour is now demonstrated with a factor of 5:

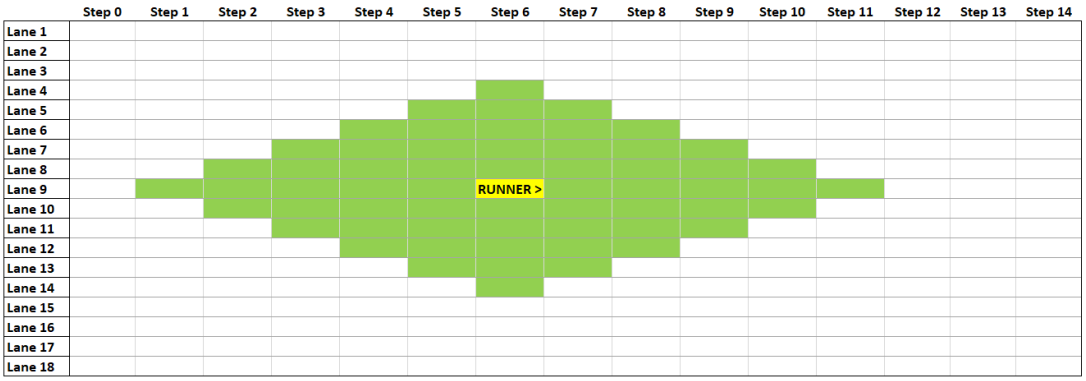

Figure 4: Runner Visibility: 5

Assuming the runner is on the 7th step of his lane, with a visibility of 5, he can perceive the track up to 5 steps ahead and behind him. This allows him to detect refreshment points, faults and other runners into this 5 steps radius: it can be thought of as a circle of visibility with centre on the runner and radius 5. In this system implementation, the runner's lane visibility is used to detect refreshment points and faults ahead, since he doesn't care anymore for the steps behind him because he will not go back. The same way, being each lane assigned to at most one runner, the visibility on his left or right is used to check for the presence of close competitors, both ahead and behind this time. Finally, if an athlete is close to the race track borders (i.e. his assigned lane is one of

the first or last ones) he will have less chances to perceive competitors compared to the middle lanes runners.

5 Self-assessment / Validation

The produced software models a self-organizing system through the use of the JaCa technology, so it is compliant to the given requirements in that

- it uses agents for autonomy and encapsulation of behaviour;
- the agents are situated, achieving local actions and perceptions;
- thanks to the CArtAgO environment management, MAS components are distributed and interact through coordination;
- in this MAS, entities cooperate and compete to reach their goals and the whole organization purpose.

The final system allows the human user to interact with and observe the race event, achieving so the semi-closed system requirement, as defined in section 2. Finally and most importantly, the ultimate result is accomplished: many runners meet for a race event and try their best, according to their internal sensations and external environmental conditions, to win the competition.

6 Deployment Instructions and Usage

In order to deploy the system, it is necessary to configure the environment by installing

1. the latest JDK version, available at the Java website;
2. the latest Gradle version, available at the Gradle website.

Next, one should download the project sources from the GitLab repository. Then, it is required to launch the command *gradle wrapper* inside the project folder in order to generate the gradlew script that invokes the declared version of Gradle, downloading it beforehand if necessary. Finally, the system can be launched with the command *gradlew runAgentRaceMas* from inside the project folder.

Once the system is running, the user can follow the event and interact with the race. To better understand how, it is possible to read the information displayed on the GUI and those printed in the console log. It is possible to change the console log verbosity by adjusting the *debug* setting in */agentrace/agentRace.mas2j*. Also, the number of spawned runners, available lanes and race length can be adjusted by modifying */agentrace/agentRace.mas2j* and */agentrace/src/main/asl/organizer.asl*.

7 Conclusions

The realized project implements a race event where many entities interact to carry out a competition. An organizer calls for a race and provides the athletes with a running track, while many runners attend the event and challenge themselves and the others to win it. Everything occurs under the eye of the human user, which can interact and modify the race conditions in order to favour or disadvantage some runners more than others, or can just observe and enjoy the race.

7.1 Future Works

The possibilities to expand this project are limitless: just consider how many more features a runner can be endowed with. Amongst those, the main improvements the system could benefit of are:

- controllable speed: in real life you have to wait until the end of the event, there's not fast forward button of course. In this case, being a software one, it could be useful for the human user to have a fader to control the overall event speed;
- better runner's perceptions: in this version of the system a runner can perceive if a competitor is close to him (i.e. it is in his visibility range), but does not distinguish if the opponent is ahead or behind him, nor cares about his actual speed;
- improved runner's reasoning: with more accurate and more information available a runner could shape differently his strategy, taking into consideration also the other runners' ones;
- other features could be added to the event, like the award ceremony, and to the track, like the presence of hills or beaches.

7.2 What did I learn

Thanks to this project I had the opportunity to deepen my knowledge and understanding of agents and autonomous systems in general. More specifically, it was helpful to learn a new reasoning and programming paradigm, which can be exploited to model many real life scenarios. Finally, I experimented with the JaCa technology and infrastructure, which opened me to new ways of thinking about projects and about their design and architectural choices.

References

[1] Alessandro Ricci, Michele Piunti, and Mirko Viroli. Environment programming in multi-agent systems: An artifact-based perspective. *Autonomous Agents and Multi-Agent Systems*, 23:158–192, 09 2011.

[2] Alessandro Ricci and Andrea Santi. Cartago by example. *Mines de Saint-Étienne*, 04 2011.

[3] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. Cartago: An infrastructure for engineering computational environments in mas. *Research Gate*, 01 2006.