Exercise Sheet 03: Architecture

Software Engineering (WiSe 24/25)

Submission: Fri. 22.11.2024, 10:00 AM
Meeting: Fri. 15.11.2024

Case Study: Robot Football

For several years, much research effort has been invested in the development of control and decision systems that interact with their environment (e.g., cyber-physical systems). Popular examples include, among others, autonomous robots and autonomous driving. To identify recurring problems and develop appropriate solutions, specific application domains have been proposed. One of these domains is Robot Football, in which two teams of autonomous robots compete against each other in a small version of a football game. There are various leagues, which are distinguished, among other things, by the type of robots participating. An example is the RoboCup Small Size League, which uses CUBE robots (see Figure 1).

Your task is to consider various architectural styles and, with their help, design and model an architecture for a simplified variant of this application domain. The architecture should enable a team of autonomous robots to play football together.

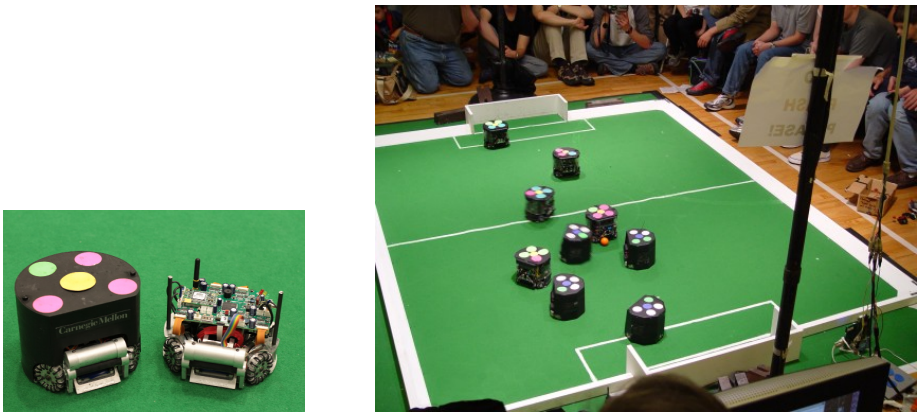

Figure 1: Example of a robot and a robot football game.

The goal of a team is to score more goals than the opposing team. Both teams have the same private resources available to them:

- Drei Robots (Note: In RoboCup, larger teams are often used). Each of the three robots on a team can be uniquely identified by a pattern of colored circles on the top. For the circles, CMYK colors (cyan, magenta, yellow, black) are used, so they cannot be confused with the playing field (green) and the game ball (orange). The robots themselves do not perform any calculations, but receive the necessary information for the game (e.g., the new target position that a robot should take) from the central team computing unit.

1https://www.robocup.org

- A central team computing unit that can communicate bidirectionally with each robot individually, as well as send broadcast messages to all robots of its own team.

Both teams also use the following shared resources:

- The playing field: all actions are carried out on a flat mat that allows smooth robot movements. The mat is uniformly green in color, except for the white markings. In particular, a white rectangle measuring 200cm x 100cm limits the playing area. The other white markings (e.g., the center circle and the goal markings) are irrelevant to the actual game action.
- Goals: The middle 40cm strip on both short sides of the rectangle is covered with an invisible sensor. Whenever this sensor detects the ball crossing the line entirely, it reports this to the central game computation unit, which records a goal for the corresponding team.
- Game ball: A small, bright orange ball is used for the game. The robots move according to the game strategy (dictated by the central team computation unit) towards the ball to shoot it into the opponent's goal as much as possible.
- Central game computation unit: This computation unit manages the game score and recognizes game events, such as: the ball leaves the playing area, or a goal is scored. All these events are reported to the team computation units simultaneously.
- A bird's-eye camera: This camera continuously captures images of the entire playing area at low resolution and forwards them to the team computation units and the central game computation unit. The camera provides four images per second, meaning one image every 250 milliseconds.

The task of the team computation unit is to determine the positions of the team players and the ball, develop a global game strategy, and inform the robots about the important part of this strategy. To determine all player positions, the team computation unit must process a camera image according to the CMYK colors with four filters (components). Accordingly, an additional (orange) filter is also needed for the ball. Such filters are available to the team computation unit. However, these filters are smaller than the actual image: they can only filter 1/3 of the full image at a time. Furthermore, each filter requires 500ms to complete all calculations.

Once the positions of the players and the ball have been recognized, a game strategy must be developed by the team computing unit, to then transmit the corresponding information (e.g., new target position for each robot) to the respective robots.

All these processes (identification of player and ball positions, strategy development, and communication of the strategy to the robots) must be continuously carried out and completed as quickly as possible.

In addition to processing the stream of images, the central team computing unit responds to events such as “goal scored” and “ball leaves the playing field,” about which it is informed by the central game unit. Predefined strategies are used for these events, e.g., the robots take up their normal starting formation after a goal.
The central game computation unit identifies the event "Goal is scored" through a message from a sensor installed in each goal that detects the ball crossing the goal line. The event "Ball leaves the playing field" is identified by analyzing the images from the bird\'s-eye view camera, which requires an orange filter to determine the position of the ball – identical to the filter used by the team computation units.

All computation units (game and team computation units) do not share resources and components, i.e., each computation unit has its own filters, for example. The computation units are not subject to any limitation of hardware resources (e.g., CPU and RAM) to enable parallel processing steps. For this purpose, components of the same type can be instantiated multiple times on a computation unit.

Task 1: Identification of the components and their functionality (39 points) Your task is to develop an architecture for these application novels, whereby they have to do a team. Therefore, the system consists of the following subsystems, diesoftware components:

- Three robots
- A central team accounting unit
- Two sensors, one for each goal of the playing field
- A central game computing unit
- A bird's-eye view camera

For the design of the architecture, consider the following three scenarios in their entirety (i.e., from start to finish):

1. “Tor!”
   Start: Ball crosses the goal line.
   End: The robots of the team take up their predefined starting positions.
2. "Ball out"
   Start: Continuous capture of images by the bird\'s-eye camera and the ball exceeds the playing field marking.
   The robots of the team take their predefined positions around the insertion point.
3. “Game Strategy”:
   Start: Continuous image capture by the bird's-eye view camera.
   End: Robots continuously receive new target positions and control these.

For each of the three scenarios:

(a) (15 Points) Description of the Scenario: Describe how to implement the scenario in a list of up to 6 steps. You may use bullet points. (5 Points per scenario)

Notes:

- For the entire task, a table for each scenario could also be offered.
- Refinement (reduction of abstraction) of partial steps should not be carried out to keep the complexity of the scenarios within limits. For example, the development of a game strategy based on the positions of the players and the ball could be considered as a partial step.

(b) (12 Points) Subsystems and Components: Assign each sub-step (a) to a subsystem that realizes it (Robot, Central Team Processing Unit, Gate, Central Game Processing Unit, Bird's Eye Camera). Identify possible components here that implement the software parts of this sub-step. (4 points per scenario)

Notes:

- Generally, a sub-step can be realized by one or more components.

(c) (6 Points) Dependencies between subsystems: This task is to be answered in Exercise 03 on Moodle. (2 points per scenario)

(d) (6 points) Non-functional requirement: This task is responsible in Moodle Ubung 03. (2 points per scenario)

Task 2: Design and modeling of the architecture (61 points) Design and model an architecture with an UML component diagram that includes sub-systems, components, interfaces and connectors (give names to all elements of the diagram). Please create a UML component diagram per scenario. Choose whether architecture styles (layers, pipes & filters, peer-to-peer, and blackboard) are applicable to sub-problems of the scenarios and if necessary use these styles in the design and modeling.
Justify your choice or non-choice of a style.

Notes:

- Refer to the guidelines for architectural modeling from the exercise.