## CAPÍTULO 2

# ARQUITETURA DE SOFTWARE PARA UM TIME DE FUTEBOL DE ROBÔS AUTÔNOMOS

**Gabriel Borges da Conceição**

**RESUMO**

A RoboCup tem o propósito de, em 2050, possuir um time de robôs humanoides totalmente autônomos capazes de jogar contra e derrotar a equipe campeã mundialde futebol humana. Entre suas diversas categorias, a SSL se destaca pelo fato de ser uma das mais populares e promover jogo dinâmico com robôs projetados eletrônica emecanicamente para realizarem chutes e inteligência artificial para realizarem jogadas conjuntas, como passes, entre outras. Este artigo tem por objetivo o estudo e o desenvolvimento prático de arquitetura de software a ser implementada na Inteligência Artificial de futebol de robôs. A arquitetura a ser concebida ao final do projeto visa alcançar aplicação em qualquer categoriade futebol de robôs, entretanto sua aplicação prática será feita na categoria F180 SSL (Small Size League), em linguagem de programação LabVIEW. Além disso, o projeto irá abranger as nuances do Diagrama de Classes concomitante à arquitetura, utilizando a ferramenta UML.

**Palavras-Chave:** RoboCup; Arquitetura de Software; Robôs autônomos; Futebol.

## INTRODUÇÃO AO JOGO

Cada time deve conter um máximo de 6 robôs, sendo um goleiro e 5 jogadores delinha. Assim como num jogo de futebol de humanos, o objetivo de cada time é fazer gols no time adversário.


Editora Epitaya | Rio de Janeiro-RJ | ISBN 978-85-94431-65-3 | 2024


Figura 1: Disposição dos robôs em campo (1)

Os robôs, em geral, possuem rodas omnidirecionais, permitindo movimento de translação e rotação simultaneamente e conseguem chutar a bola através de mecânicos, geralmente pistões descarregados por solenoides.

Os robôs possuem um código de cores em sua tampa. A cor central representa o time e pode ser azul ou amarela. As demais cores dizem respeito ao número de cada robô. Isso pode ser observado na figura 2.

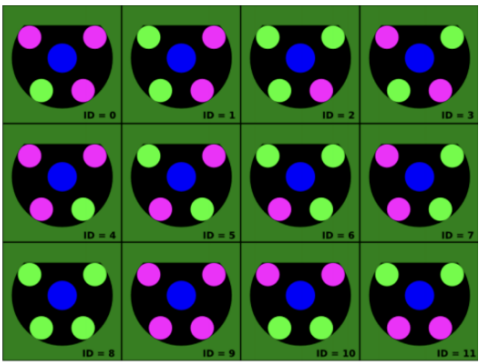


Figura 2: Disposição dos robôs em campo

## FLUXO DE INFORMAÇÃO

Primeiramente, é importante o entendimento de como se dá o fluxo das informações, recebimento de dados e algumas especificações da estrutura por trás do jogo.

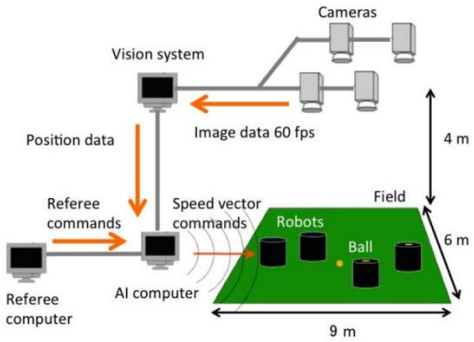


Figura 3: Fluxo das informações

Sobre o campo são dispostas 4 câmeras que rodam 60 vezes por segundo. Cada câmera é posicionada no centro de determinado quarto de campo e manda suas imagenspara o(s) computador(es) da visão, o qual é responsabilidade da organização da competição. Esse computador dispõe do ssl vision (2), programa que, a partir das imagens eda calibração as cores, as transforma em posição de robôs, bola e geometria do campo, codifica as informações e envia via rede para os computadores das equipes. Um dos computadores da organização também executa o ssl game controller (3), o qual funciona como juiz e também envia seus comandos às equipes. Com essas informações, o computador de cada equipe deve executar sua Inteligência Artificial e controlar seus robôs.

**Estruturação do código**

Antes de começar a pensar sobre a Inteligência e a lógica do jogo de fato, é importante ter bem esquematizado os módulos de execução do código e a maneira como elesse comunicam. Não é razoável, por exemplo, que o código tenha que esperar que um comando ser enviado ao robô para receber novas informações da visão.

Por isso, o código deve ser separado em módulos de execução paralela de forma a cada tarefa ter seu funcionamento independente do processamento das demais. Com isso, apresentaremos uma lista com os módulos básicos necessários ao funcionamentodo código.

**Módulo de recebimento dos dados da visão**

Como já mencionado, as informações sobre as posições dos elementos do jogo e da geometria do campo são geradas pelo programa ssl vision nos computadores da organização da competição e enviados às equipes via rede através do protocolo UDP (4), assim como os comandos do juiz.

É muito importante que se tenha um módulo dedicado apenas ao


Editora Epitaya | Rio de Janeiro-RJ | ISBN 978-85-94431-65-3 | 2024

recebimento da visão. Ter essa funcionalidade em série com qualquer outro tipo de processamento pode acarretar acúmulo de pacotes enquanto se dá o processamento. Por vezes, é comum pensar em colocar o recebimento de pacotes seguido de algum filtro para as informações das câmeras. Dessa forma, um novo pacote só poderá ser lido pelo códigoquando o filtro terminar de ser executado e isso pode fazer com que se pegue pactoes com informações defasadas já que provavelmente ocorrerá a formação de uma fila nocomputador.

Portanto esse primeiro módulo dedica-se apenas ao recebimento dos pacotes via UDP. Vale ressaltar que se recebe uma string, e deve-se transformá-la num vetor de inteiros afim de poder usar o protocolo de decodificação protobuf (5).

Neste ponto pode haver certa indecisão em relação a se decodificação deve também estar nesse módulo ou em outro. Anexar a decodificação aqui a princípio não causaria problema pois é um processamento simples. No entanto, aqui será feita a escolha de separar essas tarefas em módulos diferentes.

Detalhe importante é que a visão envia uma câmera por vezes e não o conjunto das4 câmeras. Como cada câmera trabalha a 60 fps, este módulo deve conseguir funcionara 240 fps, de forma a não se acumular ou perder pacotes.

**Módulo de decodificação e filtragem das informações**

A primeira tarefa deste módulo é utilizar o protocolo protobuf para decodificar o vetor de inteiros recebido do módulo anterior e transformá-lo de fato em posição de cada elemento e geometria do campo. Vale ressaltar que esse módulo e o anterior ocorrerão em loops paralelos, em Multithreading (6) e, portanto, é preciso pensar em como se dará a transferência de informações entre eles. Aqui será escolhida a passagemde parâmetros por referência, mas também se pode usar variáveis globais.

A segunda tarefa deste módulo diz respeito ao filtro das informações recebidas, bem como a estimação das velocidades e acelerações dos elementos no campo. Há que se saber que existem significativas imprecisões nas informações recebidas, seja por ruído, imprecisão das câmeras utilizadas, dentre outros; e isso precisa ser tratado antes de utilizar essas informações no restante do código.

Outro ponto muito importante é saber tratar o overlap das câmeras.

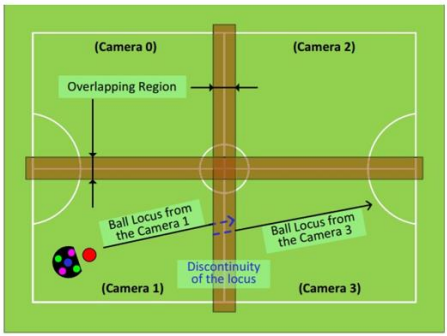


Figura 4: Overlap das câmeras

Ou seja, é possível e bastante provável receber praticamente no mesmo instante duas informações sobre o mesmo robô com posições não idênticas, pois vistas por câmeras diferentes. Vale ressaltar que no pacote da visão também vêm informações como o instante (timestamp) em que determinado frame daquela câmera foi gravado. Geralmente, utiliza-se o filtro de Kalman (7) em aplicações como essas.

**Inteligência Artificial**

Este módulo dedica-se a de fato utilizar as informações recebidas para pensar no que os robôs devem executar em cada situação. Posteriormente, neste módulo se dará aprofundamento especificando a arquitetura de Inteligência a ser implementada, a STP(21).

O objetivo desse módulo é pensar nas estratégias do jogo, gerar destino, trajetóriae velocidade para cada robô (estes dois últimos dependendo da implementação de cada equipe).

**Comunicação**

Por fim, este módulo destina-se ao envio de comandos (destino ou velocidade, de-pendendo da implementação) aos robôs e geralmente se dá via comunicação rádio. Detalhe interessante é que este módulo não necessita ser executado tantas vezes quantoo recebimento de pacotes, depende de quantas vezes por segundo a eletrônica do robôrecebe as informações.

**Diagrama de Classes**

Um dos pontos mais importantes antes de começar implementação de código é terbem estabelecido a organização das classes, já que se trata de uma arquitetura orientada a objeto. Não é obrigatório, mas é muito recomendável que se utilize uma ferramenta UML para transcrição do diagrama de classes em código, poupando muito trabalho dehierarquização entre elas, principalmente numa aplicação de futebol de robôs, na qualhá significativo número de classes envolvidas.


Editora Epitaya | Rio de Janeiro-RJ | ISBN 978-85-94431-65-3 | 2024

**Classes e suas hierarquias**

Para começar este processo, foi necessário estudar sobre construções e as relações de hierarquias existentes entre classes. Considerando conhecimento prévio acerca dos conceitos básicos de orientação a objeto, tem-se as seguintes definições (8):

**Associação** representa vínculos que são formados entre objetos durante a execução do sistema. No diagrama, é representada por meio de retas que ligam as classes envol-vidas, como exemplificado na figura 5.

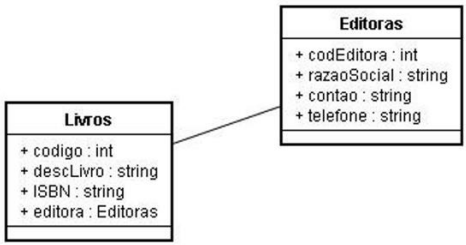


Figura 5: Representação gráfica da Associação entre classes. (9)

**Agregação** é um caso especial de associação. É utilizada para representar conexões que guardam uma relação todo-parte entre si. Numa agregação, um objeto está contidono outro, ao contrário de uma associação, como exemplificado na figura 6.

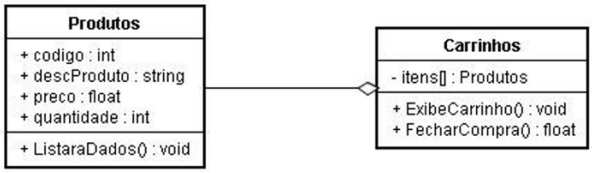


Figura 6: Representação gráfica da Agregação entre classes. (9)

**Composição** é uma variação da agregação. Na agregação, os objetos que fazem partedo todo são criados e destruídos independentemente dele. Já na composição, objetos- parte são sempre criados e destruídos pelo objeto-todo, além de os objetos-parte per- tencerem a um único todo, como exemplificado na figura 7.

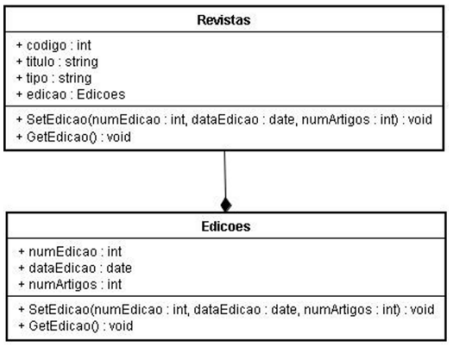


Figura 7: Representação gráfica da Composição entre classes (9)

**Generalização** se trata de um relacionamento no qual uma classe é "um tipo de"outra, onde objetos gerais se relacionam com objetos mais específicos do mesmo tipo, comoexemplificado na figura 8.

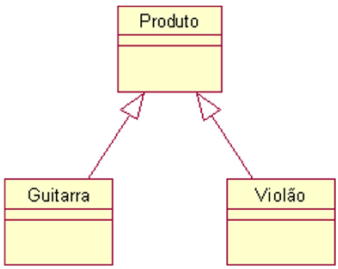


Figura 8: Representação gráfica da Generalização entre classes. (10)

Como consequência da generalização, as classes mais específicas são denominadas herdadas das classes mais gerais e possuem (herdam) todos os atributos e métodos da classe mais geral.

**Diagrama UML**

Com base nas relações entre as classes e analisando as necessidades para um código de futebol de robôs, gerou-se o diagrama contido na referência (11). Este diagrama foi construído usando a ferramenta UML do próprio LabVIEW e para ser visualizado necessita de alguma ferramenta específica de UML.

Devido a peculiaridades do LabVIEW, todos os atributos são obrigatoriamente pri-vados, portanto, estará implícito que toda classe citada possui os métodos de acesso a cada um dos seus atributos.

Dentre as classes criadas, as principais são:


Editora Epitaya | Rio de Janeiro-RJ | ISBN 978-85-94431-65-3 | 2024

**Object** é uma classe criada com o objetivo de ser a generalização para bola e robô. Seus atributos são x, y, posestx (previsão de posição que utiliza estimativas de veloci-dade), que são os atributos compartilhados pela bola e robôs.

**Ball** é uma classe herdada de Object, com apenas mais os atributos vx e vy.

**Robot** é uma classe herdada de Object com diversos atributos a mais, como orienta-tion, destx, desy, entre outros.

**Team** é uma classe construída para representar os times de robôs, possuindo como atributos a cor do time (azul ou amarelo), o lado para o qual está defendendo (esquerdoou direito), quantidade de cartões levados, entre outros.

**Ally** é uma classe herdada de Team a qual representa o time aliado. É uma outra classe pois terá métodos de input de informaçoes de velocidades aos robôs, o que não setem para o time inimigo, além de outros atributos que dizem respeito ao funcionamentodas jogadas do time, por exemplo.

**Events** é uma classe criada com o objetivo de informar os eventos do jogo, como bolafora de campo, bola dentro da área do goleiro, toque, chute, entre outros. Seus atributossão os eventos do jogo, como os citados e seus métodos são feitos para calcular a ocorrência de cada um dos seus atributos (os eventos do jogo).

**Field** é a classe que representa o campo físico. Seus atributos são informações acerca da dimensão do campo, como largura, comprimento, tamanho do gol, tamanho da área,dentre outros.

**Referee** é a classe que represnta o juiz do jogo, possuindo os atributos como comando do juiz (jogo normal, stop, falta), o id do goleiro de cada time, o tempo de jogo, etc.

**Vision** é a classe responsável por receber, decodificar e filtrar/estimar as informaçõesacerca das câmeras recebidas do computador da visão.

**Game** é a classe que possui todas as informações necessária para a IA e todas as classes citadas acima são seus atributos, além de outras mais.

Essas são as principais classes utilizadas. As demais podem ser vistas o diagrama UML informado, assim como suas relações de dependência.

**Orientação a objeto em LabVIEW**

Para elaboração prática do diagrama de classes e posterior implementação da arquitetura, é necessário realizar estudo acerca da programação orientada a objeto na linguagem utilizada, LabVIEW.

Como resultado do estudo, foram construídos os seguintes tutoriais no software degerenciamento de código:

1. Instalação do UML no LabVIEW (12)
2. Encapsulamento das classes (13)
3. Criar classe (14)
4. Adicionando atributos a uma classe (15)
5. Adicionando métodos a uma classe (16)
6. Encapsulamento dos métodos da classe (17)
7. Criar uma classe por herança (18)
8. Aplicando as classes criadas numa função main (19)
9. Utilizando passagem de parâmetros por referência em classes (20)

**Arquitetura STP: Skills, Tactics and Plays**

O comportamento do robô é dividido em uma forma hierárquica. De maneira resumida, pode-se dizer que: As Skills são as ações mínimas do robô. As Skills são selecionadas por uma Máquina de Estado que dá ordens aos robôs por meio de um conjunto de Skills. Esta máquina de estado tem o nome de Tactic. Nesse sentido, uma sequência de Tactics controla o comportamento do robô em cada Play determinada. Além disso, a Play é o componente da arquitetura que determina, através de uma sequência de Tactics para cada robô, o comportamento da equipe a qualquer momento, como um estado de passe ou uma situação para roubar a bola com o adversário.

O principal algoritmo de execução do STP pode ser visto na figura 9.

```
Process STP Execution
 1. CaptureSensors()
 2. RunPerception()
 3. UpdateWorldModel()
 4. P ← ExecutePlayEngine()
 5. for each robot i ∈ {1, ..., N}
 6.     (T_i, TParams_i) ← GetTactic(P, i)
 7.     (SSM_i, SParams_i) ← ExecuteTactic(T_i, TParams_i)
 8.     if NewTactic(T_i) then
 9.         S_i ← SSM_i(0)
10.     (command_i, S_i') ← ExecuteStateMachine(SSM_i, S_i, SParams_i)
11.     robot_command_i ← ExecuteRobotControl(command_i)
12.     SendCommand(i, robot_command_i)
```

Figura 9: STP execution algorithm (22)

**Plays**

São os elementos de mais alto nível na arquitetura STP, eles definem o comportamento estratégico da equipe. Diferente da arquitetura


Editora Epitaya | Rio de Janeiro-RJ | ISBN 978-85-94431-65-3 | 2024

implementada no STP comum, conforme especificado em (22) e (23) , no qual foi usada exatamente uma Play para toda a equipe, a versão do STP idealizada neste estudo usa três Plays simultâneas, parao KepperTeam, o DefensiveTeam e o OffensiveTeam. Ambos trabalham de forma independente para obter mais liberdade de ação durante o jogo. Do ponto de vista da programação, cada Play é uma classe herdada de uma classe base chamada Play, e cada Play possui dois métodos: um para verificar as condições de início e outro para verificar as condições de término.

As condições de término especificam quando a execução da Play deve parar; As condições de início estipulam os requisitos que o jogo deve atender para executar a Play. Ambos são baseados no comando do juiz, na posição da bola e dos robôs, dentreoutros fatores.

Um determinado estado pode satisfazer as condições de início de várias Plays ou pode existir uma situação em que duas ou mais Plays sejam igualmente prováveis. Pararesolver esse problema, foi criado um sistema de pontuação para escolher entre essas jogadas, que será explicado no próximo tópico.

A Inteligência Artificial no código é dividida em três módulos, como pode ser vistona figura 10. Cada subTeam (Kepper, Defensive e Offensive) tem sua Inteligência Artificial em paralelo com os demais.


Figura 10: Artificial Intellignece Modules

**Choose Play**

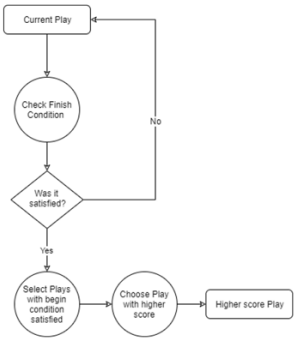


Figura 11: Choose Play Module Diagram

É o primeiro módulo do sistema de seleção das Plays e é responsável por alternar entre as Plays, se a situação do jogo exigir.

O primeiro passo é verificar se a condição de final da Play atual é satisfeita; Se não for satisfeita, a Play atual mantém sua execução. Depois que a condição de término ocorrer, o segundo passo é selecionar todas as Plays que tenham as condições de iníciosatisfeitas com o estado atual do jogo. O terceiro e último passo é escolher a Play commaior pontuação no sistema de pontuação.

O sistema de pontuação foi criado para escolher entre várias Plays com condiçõesde início satisfeitas simultaneamente e foi baseado em (23) . Cada Play tem pontuação inicial informada manualmente (hardcoded), formando uma lista de prioridades. O objetivo é que, durante o jogo, conforme as Plays sejam bem-sucedidas ou falhem, pontos sejam adicionados ou subtraídos para atualizar a lista e modificar a prioridade entre as Plays.

O diagrama do processo pode ser visto na figura 11

**Pick Robots**

É o módulo responsável por, após a Play já ter sido escolhida, coordenar o processode escolha de robôs dentre todos os robôs de toda a equipe para as sub-equipes (Kepper,Defensive e Offensive) e dentro de cada sub-equipe, associando os robôs às funções.

Inicialmente, em uma parte anterior do código, todos os robôs da equipe são classi-ficados em um vetor. Nesse ponto, as coordenadas do robô e os parâmetros da bola são usados para determinar qual deles é mais favorável para estar em uma postura ofensiva.Depois disso, o robô designado para o goleiro é colocado no topo do vetor e o restantede toda a equipe é alocado no final do vetor. Esse vetor é compartilhado com as sub-equipes usando a passagem de parâmetros por referência. Então, paralelamente, cada subequipe escolhe seus robôs.

O KeeperTeam escolhe o primeiro robô do vetor. O OffensiveTeam escolhe a quantidade necessária a paritr do início do vetor, excluindo o robô escolhido pelo Keeper-Team; a quantidade de robôs para o OffensiveTeam depende de qual Play está em execução e é um atributa de cada uma das Plays. A DefensiveTeam escolhe todos os robôsdo vetor, exceto o escolhido pelos KeeperTeam e OffensiveTeam. As informações sobre quais robôs cada subequipe escolheu são passadas por referência às três subequi-pes. Como o processo descrito é feito em paralelo, pode acontecer, por exemplo, que quando o DefensiveTeam for escolher seus robôs, o OffensiveTeam ainda não tenha es-colhido, portanto, o DefensiveTeam escolherá incorretamente, mas ocorrerá apenas por algumas poucas iterações até que o OffensiveTeam escolha seus robôs. Depois, cada subequipe associa cada robô a uma Role. Uma Role consiste em comportamentos parao robô executar em sequência, que são as Tactics, e eles serão executados simplesmenteaté que uma das condições de acabamento se apliquem.


Editora Epitaya | Rio de Janeiro-RJ | ISBN 978-85-94431-65-3 | 2024

**Execute Play**

É o módulo responsável, após a escolha de uma Play e cada robô já estar associadoa uma Role, para executar adequadamente o conteúdo da Play e chamar a execução dasRoles de cada robô.

A arquitetura STP exige que todos os robôs avancem para sua próxima Tactic simultaneamente. Como exemplo, considere o jogo de passe: um dos robôs se posicionapara executar o passe e o outro se posiciona para receber o passe. Quando esses robôsjá tiverem se posicionado, ambos avançam para sua respectiva próxima Tactic. Assim,o primeiro robô executa o passe enquanto o outro faz ajustes na sua posição para garantir que está na direção da bola. Quando o robô receptor recebe a bola, ambos avançampara sua respectiva próxima Tactic. Assim, o primeiro robô se posiciona em posição estratégica para acompanhar o ataque quando o outro chuta a bola para tentar fazer umgol.

**Tactics**

São máquinas de um estado de Skills. Como dito na subseção de Plays, quando uma Play começa, cada robô recebe uma sequência de Tactics (essa sequência é a Role).

Por exemplo, na Play de passe, um dos robôs tem uma sequência com duas Tactics:WaitPass (tática responschutar a bola em direção ao gol após ter a bola sob controle) . Enquanto isso, o outro robô tem uma sequência com duas Tactics: KickToReceiver (Tactic responsável por executar o passe para o robô receptor) e FollowAttacker (Tacticresponsável por posicionar o robô, depois de ele ter realizado o passe, em localização estratégica perto do robô com a Tactic Shoot)

Nesta Play, um terceiro robô tem uma sequência com uma única Tactic: PassOb- server (tática responsável por posicionar esse robô em um local estratégico para atrairrobôs inimigos, a fim de reduzir o número de inimigos próximos ao robô que vai rece-ber a bola e tentar marcar o gol).

Do ponto de vista da programação, cada Tactic é uma classe com apenas um método cujo objetivo é executar a Skill State Machine. Então, nesse método de cada Tactic, não há nenhuma lógica da decisão ou ação do robô explicitamente, existe apenas a operação da Skill State Machine: chamada da execução da Skill e lógica de transição de estados.

Vale ressaltar que todas as Tactics são livres, ou seja, uma Tactic pode ser inseridaem várias sequências de Tactics (Roles).

**Tactics Management**

É um ponto importante nessa arquitetura. Um robô apenas avança para a próxima Tactic em sua sequência, se todos os outros robôs da Play atual também puderem. Essaé uma maneira de coordenar todas as ações dos robôs na execução da Play.

Para fazer isso, após cada iteração da Tactic de cada robô, o robô informa o número da Tactic em sua sequência que está habilitado para

executar. Esse número pode ser *i*(número da Tactic atual, ou seja, a Tactic atual ainda não terminou), *i* + 1 (número da próxima Tactic, ou seja, a Tactic atual já está concluída) ou 1 (a Tactic atual já está concluída e é a última Tactic na sequência deste robô).

De todos os números diferentes de 1 informados pelos robôs, escolhe-se o menor e esse número é o índice da Tactic para a próxima iteração de código para todos os robôs. Na primeira iteração, todos os robôs são inicializados na Tactic 0.

Para cada robô, se o índice da Tactic informado para a iteração atual for maior que o último índice de Tactic, esse robô executará sua última Tactic. Isso ocorre quando dois ou mais robôs têm um número diferente de Tactics; se um robô tiver menos Tactics do que outro, esse robô continuará executando sua última Tactic, enquanto o outro avançará em sua sequências de Tactics.

**Skills**

São as ações do robô, elas são o componente da estrutura que efetivamente executacálculos e lógica para definir a posição do destino do robô e determina quando o robôativará o chute ou o drible, etc.

Algumas Skills são unidas para criar a Skill State Machine (Tactic). Para esclarecer, considere a Play Duelist (uma Play Defensiva na qual um robô de defesa vai até a bola para afastá-la de perto da área). Nesta Play, um robô recebe uma sequência com uma Tactic: Shhot. Essa Tactic tem dois estados: KickToGoal (o robô vai para a bola e posicionado para chutar em um ponto calculado da meta) e StealBall (o robô vai para a bola com o chute ativado, mas seu posicionamento é calculado para bloquear um possível chute inimigo). A escolha desses estados depende das distâncias do inimigo e dos aliados da bola.

Do ponto de vista da programação, cada Skill é uma classe com um método principal que gera a posição de destino do robô e possíveis métodos auxiliares dentro deste método principal para subdividir a lógica e os cálculos.

## CONCLUSÃO

Por fim, este relatório final teve como objetivo finalizar o estudo promovendo umaarquitetura de software a ser implementada na Inteligência Artificial. Juntamente com os demais módulos idealizados nas fases anteriores do estudo, foi possível obter a idealização do código desde o recebimento dos pacotes da visão até o envio de comandosaos robôs.


Editora Epitaya | Rio de Janeiro-RJ | ISBN 978-85-94431-65-3 | 2024

## REFERÊNCIAS


1. https://tinyurl.com/yynx49sk
2. https://tinyurl.com/y5at8uw3
3. https://tinyurl.com/y2mavvqr
4. https://tinyurl.com/ws3c6v6
5. https://tinyurl.com/ouz2ak9
6. https://tinyurl.com/bu82o4y
7. https://tinyurl.com/uu3q5ow
8. https://tinyurl.com/classesref
9. https://tinyurl.com/classesref1
10. https://tinyurl.com/classesref2
11. https://tinyurl.com/su9oj5b
12. https://tinyurl.com/umllabview
13. https://tinyurl.com/encapsulamento
14. https://tinyurl.com/whr4plw
15. https://tinyurl.com/uel9ckk
16. https://tinyurl.com/rmzchn7
17. https://tinyurl.com/uvd7mh6
18. https://tinyurl.com/skba8gy
19. https://tinyurl.com/r8jnnce
20. https://tinyurl.com/rkcp3ep
21. B Browning*, J Bruce, M Bowling, and M Veloso: STP: skills, tactics, and playsfor multi-robot control in adversarial environments. Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 2003 https://tinyurl.com/wjdqol
22. author = B. Browning, J. Bruce, M. Bowling, M. Veloso, Title = STP: skills, tactics, and plays for multi-robot control in adversarial environments, School = Carnegie Mellon University, Pittsburgh, Pennsylvania, USA , Pages = 1-20, howpublished ="http://papersdb.cs.ualberta. ca/~papersdb/uploaded_files/556/paper_p33.pdf", note = Avai-lable at http://papersdb.cs.ualberta.ca/~papersdb/uploaded_
23. author = Christian K¨onig, Daniel Waigand, Florian Schwanz, Gunther

Berthold, Malte Mauelshagen, Tobias Kessler, Title = TIGERS Man- nheim, Atificial Inteligence, School = Department of Information Te- chnology, Department of Mechanical Engineering Baden-WuerttembergCooperative State University, Coblitzallee 1-9, 68163 Mannheim, Ger- many, Pages = 10-17, howpublished https://tigers-mannheim. de/download/papers/2011-AI-Structure_Koenig.pdf, note = Available at https://tigers-mannheim.de/download/papers/


Editora Epitaya | Rio de Janeiro-RJ | ISBN 978-85-94431-65-3 | 2024