Especificando Comportamentos para Futebol de Robôs da Abordagem DDM

Raoni Sales de Oliveira[1], **Ana Patricia Fontes Magalhães Mascarenhas**[1]

[1]Universidade Salvador (UNIFACS)
Salvador, BA – Brazil

Abstract. *Robot soccer is often used as a test bed for Multi-Agent Systems (MAS). In a soccer team, each player is an agent, and his actions should be coordinated with other agents to reach a goal. In this context, coordinated plans, called Setplays, are used to simulate and plan real game situations. Tools for defining Setplays usually have a limited set of behaviors available for use, e.g. kicking, and new behaviors have to be hand-coded. This work proposes a solution to support the implementation of behaviors for simulation and execution of cooperative MASs, in the context of robot soccer. The solution uses Model Driven Development (MDD), which allows the definition of behaviors at a high abstraction level and subsequent (semi) automatic transformation into source code for specific tools. Experiments with robot soccer teams demonstrated the viability of the solution in defining behaviors and generating source code for the FCPortugal Framework and SPlanner tools used by several robot soccer teams.*
Keywords: *MAS, behavior, Robotic soccer, MDD, metamodel, code generation.*

Resumo. *O futebol de robôs é frequentemente utilizado como berço de teste para Sistemas Multi Agentes (SMA). Em um time de futebol, cada jogador é um agente, e suas ações precisam estar coordenadas com outros agentes para alcançar um objetivo. Neste contexto, planos coordenados, denominados setplays, são utilizados para simular e planejar situações reais do jogo. Ferramentas para definir setplays geralmente possuem um conjunto limitado de comportamentos disponíveis para uso, ex. chutar, e novos comportamentos precisam ser codificados manualmente. Este trabalho propõe uma solução para apoiar a implementação de comportamentos para simulação e execução de SMAs cooperativos, no contexto de futebol de robôs. A solução utiliza o Desenvolvimento Dirigido a Modelos (DDM), que permite a definição de comportamentos em alto nível de abstração e subsequente transformação (semi) automática em código-fonte para ferramentas específicas. Experimentos com times de futebol de robôs demonstraram a viabilidade da solução na definição de comportamentos e geração de código-fonte para as ferramentas FCPortugal Framework e SPlanner utilizadas por diversos times de futebol de robôs.*
Palavras-chave: *SMA, comportamento, futebol de robôs, DDM, metamodelo, geração de código.*
Dissertação de Mestrado apresentada em 20/10/2021.

1. Introdução

Sistemas Multi-Agente (SMA) é um ramo da Inteligência Artificial (IA) onde agentes realizam tarefas distribuídas em um sistema cooperativo [Russell and Norvig 2020]. Em

SMA, os robôs são considerados agentes autônomos dotados de inteligência. Podem fazer planos e inferências estabelecendo comunicação mútua [Konolige and Nilsson 1980]. IA e SMAs tem sido adotadas em diversas aplicações, e para fomentar pesquisas nestas áreas o futebol de robô tem sido usado como berço de teste.

É comum no desenvolvimento de SMAs a utilização de simuladores que possibilitam imitar a execução do sistema e observar o seu comportamento. Existem ferramentas de simulação para SMAs de diferentes domínios [Andrade et al. 2016][Pessin et al. 2007][Jung et al. 2019][Hogg and Sretavan 2005].
No domínio do futebol de robôs, simuladores podem ser utilizados, por exemplo, para a definição e execução de setplays, ações coordenadas multiagente que expressam o esforço conjunto de uma equipe em obter um resultado em comum. Em analogia ao futebol de seres humanos, um *setplay* pode ser visto como uma jogada ensaiada (JE) entre os participantes de um time. Turtle Simulator [Meessen 2011], SimRobot [INFORMATIK 2011] e SPlanner [Cravo 2011] são exemplos de ferramentas usadas no domínio do futebol de robôs.

As ferramentas de simulação para *setplays*, em geral, possuem um conjunto limitado de comportamentos disponíveis para serem utilizados. Um comportamento pode ser visto como uma ação atômica que pode ser executado por um agente. No domínio do futebol de robôs, correr, driblar e chutar são exemplos de comportamentos atômicos que um agente pode executar em campo. O aperfeiçoamento dos *setplays* com novos comportamentos exige o desenvolvimento manual na ferramenta de simulação adotada. Portanto, requer conhecimento dos códigos-fonte específicos e um grande esforço de programação, além de deixar o comportamento acoplado e dependente de uma ferramenta.

Existem abordagens de desenvolvimento de software que diminuem a complexidade do desenvolvimento multiplataforma através de abstrações, dentre essas abordagens está o Desenvolvimento Dirigido a Modelos (DDM) [Brambilla et al. 2012]. DDM utiliza linguagens de modelagem específicas de domínio (DSLs) para modelar um sistema. Estes modelos são independentes de plataforma e podem ser convertidos de forma (semi) automática para a linguagem nativa de múltiplas plataformas dentro de um mesmo domínio.

Esta dissertação propõe o uso da abordagem DDM para a construção de novos comportamentos no contexto de futebol de robôs simulados. Para isso, define a solução *Robot Soccer Behavior Generator (RoboSocBG)*, para modelar novos comportamentos de forma independente de plataforma. Nosso objetivo é dissociar o conhecimento inerente ao novo comportamento da tecnologia utilizada para sua implementação. O RoboSocBG compreende: (i) uma linguagem de modelagem para o domínio de futebol de robôs simulados; (ii) uma cadeia de transformações que mapeia modelos de comportamento em código; (iii) uma ferramenta automatizada para definir comportamentos em um alto nível de abstração e gerar código em plataformas específicas.

A partir de uma linguagem de modelagem específica para o domínio de futebol de robôs, modelos de comportamento instanciados podem ser reusados na geração de código de forma (semi) automática para diferentes plataformas alvo. Atualmente RoboSocBG gera código para as ferramentas FCPortugal Setplays Framework (FSF) [Mota et al. 2015] e SPlanner [Cravo et al. 2014]. RoboSocBG foi testada em nosso laboratório e validada em um estudo de caso, e se mostrou viável para geração de código

nestas duas plataformas. Outras ferramentas serão adicionadas futuramente.

A seção 2 apresenta o domínio de futebol de robôs, as soluções atuais relacionadas à criação de comportamentos para *setplay*, além de introduzir a abordagem DDM. Em seguida, a Seção 3 apresenta a solução RoboSocBG detalhando como ela foi definida. A seção 4 mostra o ambiente que automatiza o RoboSocBG e a Seção 5 demonstra a sua validação. Finalmente na Seção 6 são apresentadas as conclusões e trabalhos futuros.

2. Background

A Copa do mundo de Robôs (RoboCup) tem a intenção de utilizar o futebol na promoção de ciência e tecnologia, e assim tratar grandes desafios em inteligência artificial de forma mais atraente ao público [Robocup]. A RoboCup contém diversas ligas de competições, dentre elas a RoboCup 3D Soccer Simulation, que trabalha com robôs simulados. Nesta, robôs autônomos (agentes) interagem e cooperam em um time de futebol, compondo um SMA, para atingir um objetivo em comum, vencer o time adversário.

Robôs podem ser treinados através de jogadas ensaiadas, chamadas *setplays*. Um *setplay* pode ser visto como uma sequência de comportamentos executados pelos jogadores para atingir um objetivo em comum. Nesta dissertação, definimos cada ação que compõe um *setplay* como um *comportamento atômico*. O *setplay Cobrar um escanteio*, por exemplo, pode envolver um jogador *chutando a bola* e outro jogador *cabeceando a bola* no o gol, ou seja, dois comportamentos atômicos, *chutar a bola* e *cabecear a bola*.

A especificação manual de *setplays* com mais de quatro etapas pode ser uma tarefa difícil, com grande possibilidade de gerar erros [Reis et al. 2010], pois um *setplay* é uma máquina de estados, e a sua implementação envolve a sincronização entre os agentes para cada mudança de estado. Codificá-lo manualmente, além de ser susceptível a erros, requer uma longa etapa de depuração. Então, é comum utilizar ferramentas para auxiliar a criação de *setplays*.

2.1. Soluções que apoiam a criação de setplays

Esta seção apresenta o estado da arte em ferramentas para criação de *setplays* e propostas para definição de comportamentos. Para identificar as ferramentas usadas na criação de *setplays* foi conduzido um *survey* com os times da subliga 3D Soccer Simulation da RoboCup. Para identificar as soluções que definem comportamentos, foi realizada uma Revisão Sistemática da Literatura (RSL).

O *survey* identificou as seguintes ferramentas: FCPortugal Setplays Framework [Mota et al. 2015], SPlanner [Cravo et al. 2014], MagmaDeveloper, Matchflow [Marques 2010], Playmaker [Reis et al. 2010] e uma ferramenta proprietária.

Para a RSL que investiga os trabalhos que desenvolvem novos comportamentos foram utilizadas as bases de dados ACM e IEEE, considerando artigos publicados a partir de 2009 e escritos em inglês. A string de busca adotada está especificada a seguir: *((multi-agent OR MAS OR multi agent) AND (MDD OR model-driven OR MDE OR MDA OR model OR specification OR development OR implement) AND (BEHAVIOR OR BEHAVIOUR OR coordination OR planning OR simulation))*.

A busca retornou 2945 artigos. Destes, 105 foram selecionados a partir do título, 23 com a leitura do resumo e, 8 artigos foram analisados na íntegra (Tabela 1).

Tabela 1. Artigos seleciondos na revisão sistemática de literatura.

Id	Ano	Título	Base
1	2011	A model design of multi-agent systems	IEEE
2	2011	Making Multiagent System Designs Reusable: A Model-Driven Approach	IEEE
3	2012	Model-Driven Behavior Specification for Robotic Teams.	ACM
4	2012	Research on the Behavior Model of Enterprise Network Agent	IEEE
5	2013	Petri Net based agent behavior definition and execution for Multi-agent Systems	IEEE
6	2017	Model-Driven Engineering in Agent-based Modeling and Simulation: a Case Study in the Traffic Signal Control Domain	ACM
7	2018	New Approach of Designing and Developing Multi-Agent Systems	ACM
8	2020	Strategy Planner: Enhancements to support better defense and pass strategies within an LfD approach	IEEE

A partir da análise dos artigos constatou-se que somente a solução apresentada em [Paraschos et al. 2012] está no contexto de futebol de robôs. Contudo, oferece suporte à criação de ações coordenadas multiagente (ou comportamentos de equipe), e não aborda a problemática da criação de comportamentos atômicos.

A falta de referências na literatura para soluções que deem suporte a modelagem de comportamentos atômicos demonstra que essa etapa de programação dos agentes SMA ainda é realizada de forma manual, complexa e passível de erros de codificação.

2.2. Desenvolvimento Dirigido por Modelos (DDM)

A proposta do DDM é construir modelos em alto nível de abstração, e convertê-los de forma (semi) automática através de uma cadeia de transformação em outros modelos, até gerar o código fonte [Brambilla et al. 2012].

O DDM compreende os modelos, que representam um sistema em diversos níveis de abstração, e as transformações, programas que (semi)-automatizam o processo de desenvolvimento convertendo os modelos em código. A geração de código utiliza abordagens espeíficas, tais como o template-based [OMG 2008]. Neste, modelos podem ser convertidos em artefatos textuais, como códigos e documentações. Para isso, um conjunto de templates é parametrizado com os elementos de um modelo. Cada template especifica um texto contendo espaços reservados para dados a serem extraídos do modelo.

3. Solução RoboSocBG

A solução Robot Soccer Behavior Generator (RoboSocBG) utiliza DDM para otimizar a criação de comportamentos atômicos para definição de *setplays*. Os novos comportamentos são modelados em alto nível de abstração, independentes de plataforma, e utilizados para geração (semi) automática do código em ferramentas específicas. A modelagem do comportamento de forma abstrata e independente de plataforma é realizada com base em

um metamodelo, que contém os conceitos relevantes para o domínio de futebol de robôs. Assim, um modelo (instância do metamodelo) definido para um novo comportamento é utilizado na geração do código para ferramentas específicas. Atualmente o RoboSocBG gera código para as ferramentas FSF e SPlanner.

A solução proposta compreende o metamodelo com os conceitos do domínio de comportamentos, e um conjunto de transformações para automatizar o processo de geração de código para plataformas específicas. As subseções a seguir descrevem o processo de especificação do metamodelo 3.1, e o metamodelo propriamente dito 3.2. Em seguida, a Seção 3.3 apresenta a definição das transformações.

3.1. Processo de especificação do metamodelo

Definir metamodelos é uma tarefa complexa, pois em geral não existem usuários para realizar o levantamento dos requisitos do domínio. Uma estratégia para definir esses requisitos é a utilização de modelos, aplicações, taxonomias, ou outros tipos de artefatos que ilustrem de forma concreta o domínio e possam ser utilizados como referência [Magalhães Mascarenhas et al. 2015].

Neste trabalho utilizamos como artefatos para o levantamento dos requisitos as ferramentas já existentes de simulação e suporte à definição de *setplays*. As ferramentas SPlanner e FSF foram adotadas como referência por serem amplamente utilizadas pela comunidade. Inicialmente foram identificados os comportamentos presentes em ambas as ferramentas e em seguida foi realizada uma inspeção no código dos seus comportamentos.

Alguns dos comportamentos identificados foram *Pass*, passe de bola para outro jogador; *Shoot*, chute a gol; e *Run*, movimento em campo sem bola. A inspeção dos códigos do comportamento teve como objetivo observar os atributos definidos, métodos e parâmetros utilizados, estruturas de controle (ex. condições) e padrões de código reutilizados em diferentes comportamentos, que pudessem ser utilizados na geração de novos comportamentos. Por exemplo, na ferramenta SPlanner, a construção de uma jogada ensaiada é realizada por meio de uma sequência de ações executadas por jogadores. A Figura 1 mostra o trecho do código que representa essa estrutura. A inspeção deste código levou a identificação do conceito de *ação (action)*, *campo (field)* e *jogador (player)* e das características destes conceitos, como o tipo da ação (*type*), um código único que cada comportamento possui. Adicionalmente, os diagramas de classes do SPlanner e do FSF também foram analisados e comparados.

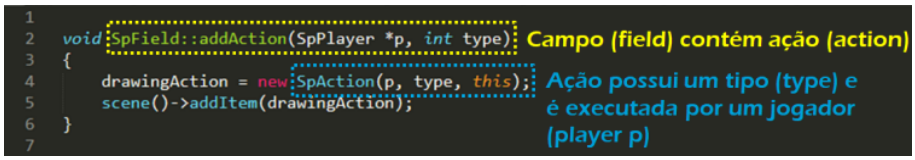

Figura 1. Trecho de código inspecionada da ferramenta SPlanner

Como resultado da análise dos códigos e diagramas das ferramentas foram mapeadas as características que definem os comportamentos já existentes, classificadas em 3 grupos: Grupo 1, características estruturais, ex. nome e descrição; Grupo 2, características específicas, ex. se o comportamento realiza passe (*Pass*) ou chute à gol (*Throw the ball in the goal*); e Grupo 3, condições especiais de execução, ex. em um escanteio o jogador não poderá sair driblando os adversários e chutar a bola no gol.

3.2. Metamodelo do RoboSocBG

O metamodelo definido para o RoboSocBG é apresentado na Figura 2. As metaclasses de cor branca não são relevantes ao processo de criação de novos comportamentos, no entanto foram representadas para complementar os elementos de um setplay. A metaclasse *BgAction* representa o comportamento e define conceitos abstratos que modelam os comportamentos que um jogador pode executar, como driblar, receber a bola e marcar. A metaclasse *BgDiretives* representa as ações que o jogador deve ou não executar em um passo (*BgStep*) e pode ser dos tipos *do* ou *dont* (atributo *type*). *BgDiretives* define os conceitos abstratos utilizados para alocar comportamentos em uma jogada ensaiada.

Os passos de uma jogada ensaiada se relacionam entre si por meio de transições (*BgTransition*). Transições possuem conditions que devem ser satisfeitas para dar continuidade ao *setplay*. Transições podem ser dos tipos *Abort, Finish* ou *NextStep*. *Abort Transitions* representam situações em que o *setplay* precisa ser abandonado, *Finish Transitions* indicam que o *setplay* atingiu seu objetivo e deve ser finalizado, e as *Next Step Transitions* são as transições que realizam a ligação entre os passos de uma jogada.

BgPorting define os conceitos abstratos utilizados para modelar os mecanismos de importação e exportação de jogadas ensaiadas, que são atualizados com os novos comportamentos gerados pela solução. Cada passo de uma jogada ensaiada possui jogadores participantes (*BgPlayer*), que são posicionados em determinadas regiões do campo (*BgPosition*) para realizar um comportamento. O comportamento de um jogador pode se relacionar com outro jogador ou com uma região do campo, *BgPlayer* e *BgPosition* possuem os conceitos abstratos para modelar esse relacionamento. Por fim, *BgField* corresponde aos conceitos abstratos que representam o campo, e *BgSetplay* compreende todos os elementos pertencentes a uma jogada.

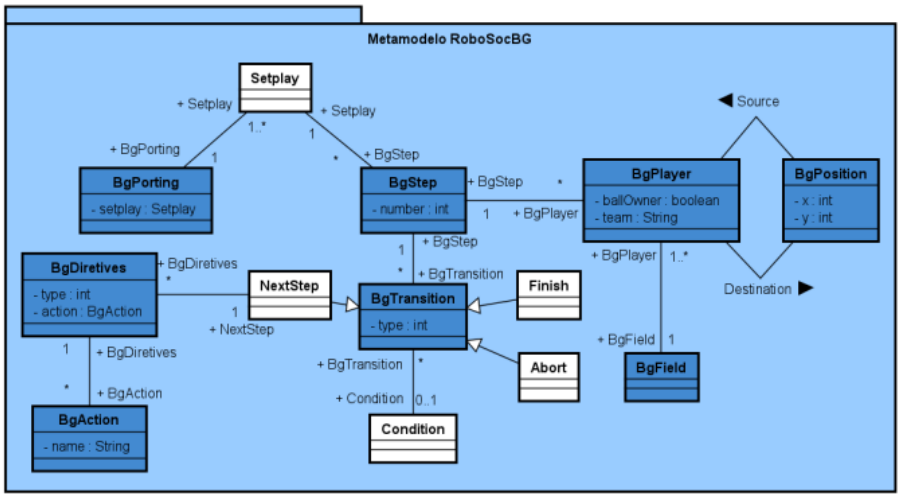

Figura 2. Metamodelo do RoboSocBG

3.3. Transformações

Transformações definem as regras para a geração de código, isto é, contém a lógica definida para conversão de um modelo de comportamento em linhas de código fonte. No RoboSocBG as transformações foram definidas usando o mecanismo de template

[OMG 2008]. A Figura 3 apresenta um trecho do template que gera código fonte para o arquivo SPPlayer.cpp. Estas regras verificam se a lógica criada para o novo comportamento usa o parâmetro *throw the ball in the goal*, se sim, utiliza os códigos de um comportamento de chute a gol para criar o novo comportamento *codeGoal*, caso contrário, utiliza os códigos de um comportamento definido no parâmetro *destination (codeDestination)*.

```
1
2   if (strcmp($_SESSION['destpoint'], "no")==0
3    || strcmp($_SESSION['goal'], "yes")==0) {
4       $code .= $codeGoal;
5   }
6   else if (strcmp($_SESSION['destpoint'], "yes")==0
7         && strcmp($_SESSION['goal'], "no")==0){
8       $code .= $codeDestination;
9   }
10
```

Verifica se é um comportamento de chute a gol

Figura 3. Template de geração de código

4. Ferramenta RoboSocBG

Essa seção apresenta a ferramenta que automatiza a solução RoboSocBG. A Figura 4 (A) fornece uma visão geral dos principais elementos da solução RoboSocBG: a interface de modelagem (*Modeling Interface*), para apoiar a definição de novos comportamentos (*Behavior Model*) em um alto nível de abstração; e o gerador de código (*Code Generator*), composto de *Transforamtion Rules*, que produzem o código fonte necessário de acordo com o modelo criado para o comportamento, e o integra às ferramentas FSF e SPlanner. A solução foi implementada utilizando a linguagem de programação Php.

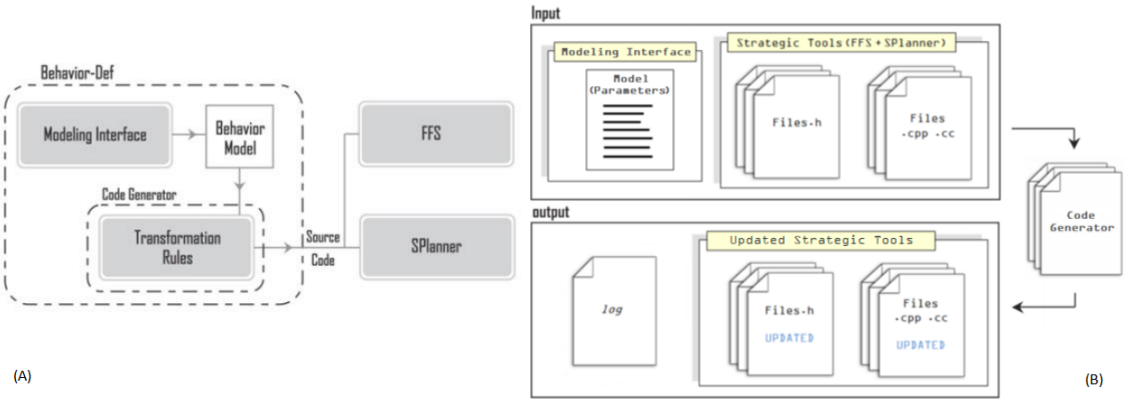

Figura 4. (A) Visão geral e (B) Arquitetura da ferramenta RoboSocBG

A ferramenta RoboSobBG contém dois requisitos, especificação de comportamento e geração de código. O desenvolvedor define um novo comportamento especificando características básicas, como o nome do comportamento e uma breve descrição, além da configuração desejada, ex. se o comportamento usa a bola ou passes. Em seguida, o comportamento especificado é usado como entrada para gerar código.

A arquitetura da ferramenta está representada na Figura 4 (B). *Modeling Interface* representa a interface de modelagem da solução, e é responsável pela coleta dos

parâmetros (*Parameters*) informados pelo desenvolvedor do comportamento. Os dados coletados (*Model*) nessa interface são passados ao *Code Generator*, onde o novo comportamento é gerado de acordo com o modelo criado pelo desenvolvedor. O gerador converte o modelo em código fonte utilizando um conjunto de regras de transformação, que são processadas nos *templates* de geração de código. Um arquivo de *log* é gerado contendo informações sobre os códigos inseridos e atualizados nos arquivos pertencentes ao FSF e ao SPlanner. Foi definido um conjunto de *templates*, cada um correspondendo a um arquivo de código fonte no FSF e SPlanner. O módulo de geração de código acessa os arquivos em suas pastas originais e os modifica sem intervenção do usuário. Um total de 9 arquivos são alterados na geração de código para essas duas plataformas.

4.1. Interface da ferramenta

A modelagem do comportamento é realizada através de uma interface gráfica provida pela ferramenta (Figura 5). A interface é dividida em três blocos: o primeiro compreende campos para definir as características básicas do comportamento; o segundo define o tipo do comportamento, ex. se será realizado com ou sem posse de bola e como é definido o destino da ação, ou seja, se é um jogador ou uma região do campo; e o terceiro determina as condições que o comportamento deverá respeitar para ser executado. O desenvolvedor pode selecionar a combinação de parâmetros que desejar para atender à lógica de funcionamento do comportamento que será criado.

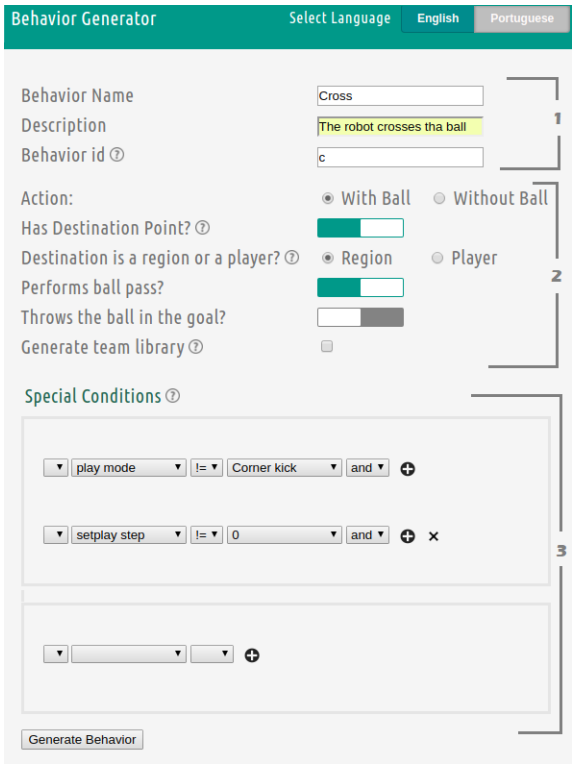

Figura 5. Interface do RoboSocBG

5. Avaliação RoboSocBG

A avaliação do RoboSocBG compreende: testes em laboratório (Seção 5.1), para avaliar a qualidade do código gerado; e estudo de caso (Seção 5.2) com equipes da RoboCup, para avaliar a viabilidade do uso do RoboSocBG na geração de novos comportamentos.

5.1. Teste

O objetivo do teste foi verificar se a ferramenta é capaz de gerar código para comportamentos já existentes e se o código gerado funciona conforme o comportamento original da ferramenta. Desta forma, os comportamentos existentes nas ferramentas SPlanner e FSF foram recriados no RoboSocBG e depois comparados com o comportamento originalmente implementados. Os seguintes passos foram executados: (i) criação de novos comportamentos pelo pesquisador utilizando a ferramenta; (ii) inspeção do código dos comportamentos criados comparando-os com o código já existente na ferramenta; e (iii) testes nos comportamentos gerados.

Após a geração de comportamentos similares aos originais do FSF e SPlanner utilizando o RoboSocBG, seus códigos foram inspecionados, compilados e executados em uma jogada ensaiada. Foi registrado a quantidade de Linhas de Código (LOC) gerada e o resultado do teste, se atingiu a saída esperada (sucesso) ou se falhou (Tabela 2). O teste foi considerado bem sucedido se o comportamento teve execução similar ao nativo da ferramenta. Como podem ser observados na tabela, todos os testes tiveram sucesso e a ferramenta foi considerada estável para ser utilizada em estudos de caso.

Tabela 2. Resultado dos testes realizados no RoboSocBG.

Comp.	Saída esperada	LOC	Resultado
Pass	Ação de passe para um jogador do time	260	Successo
Pass Forward	Ação de passe para uma região do campo	275	Successo
Shoot	Ação de chute à gol	173	Successo
Dribble	Ação que movimenta o jogador no campo com posse de bola	190	Successo
Run	Ação que movimenta o jogador no campo sem posse de bola	197	Successo
Wait	Ação sem bola e sem movimentação em campo	183	Successo
Hold	Ação com posse de bola e sem movimentação em campo	191	Successo

5.2. Estudo de Caso

O objetivo do estudo de caso foi definido usando o template GQM [Solingen et al. 2002] e consistiu em *Analisar a solução RoboSocBG com propósito de avaliar a cobertura do código gerado e o esforço de desenvolvimento com respeito as linhas de código incluidas/modificadas nas ferramentas FSF e SPlanner, na perspectiva do desenvolvedor de software no contexto de times da subliga 3D de futebol de robôs simulados da RoboCup*.

Com base no objetivo, as seguintes questões de pesquisa foram definidas: Q1: O código gerado é correto em relação ao comportamento especificado? Q2: O código gerado foi suficiente para implementar o comportamento desejado?

Para avaliar as questões de pesquisa, definimos as métricas M1, cobertura do código, e M2, correção do código. M1 corresponde a razão entre as LOC gerado e LOC total. Quanto mais próximo M1 estiver de 1 mais completo será o código gerado. Portanto, M1 = 1 indica uma cobertura de 100%, ou seja, todo o código foi gerado. M2 corresponde a razão entre casos de teste executados corretamente e o total de casos de

teste. Quanto mais próximo M2 estiver de 1 mais correto é o código gerado. Portanto, M2 = 1 indica que o comportamento obteve êxito em todos os testes realizados.

O cenário do estudo consiste em desenvolver um novo comportamento utilizando a solução RoboSocBG. O novo comportamento deve ser devidamente compilado e executado no SPlanner e no FSF. Adicionalmente, espera-se testá-lo em um *setplay*, executado pelo time BahiaRT em uma partida de futebol de robôs.

O estudo foi desenvolvido virtualmente com participantes de duas equipes da subliga 3D de futebol de robôs simulados da RoboCup mundial. Dois métodos de coleta de dados foram utilizados no estudo, a coleta indireta e a independente. A coleta indireta foi realizada através da aplicação de um questionário ao final da execução do estudo com o propósito de medir a percepção sobre a utilidade da ferramenta. A coleta independente foi realizada através da análise dos códigos produzidos pelos participantes. O código foi testado a partir dos casos de teste definidos pelo pesquisador.

Foram realizadas diversas replicações de forma individual, uma para cada participante do estudo de caso. Os seguintes artefatos foram produzidos e entregues a cada participante: apresentação da solução RoboSocBG; apresentação do cenário do estudo de caso; ambiente Linux (Ubuntu) configurado para utilizar as ferramentas e o RoboSocBG; um exemplo de comportamento para ser criado no estudo de caso; questionários on-line para serem respondidos durante as etapas do estudo de caso. O mesmo cenário foi adotado para todos os participantes.

Foram executados o estudo de caso piloto e o principal. O piloto teve o objetivo de identificar falhas no projeto do estudo de caso principal. Para isso, selecionamos dois alunos de um grupo de pesquisa com atividades relacionadas ao projeto da solução RoboSocBG. Por limitação de espaço não apresentamos o estudo piloto neste artigo.

O estudo principal foi realizado ao longo de 15 dias, com um total de 5 replicações, com integrantes de 2 times de futebol de robôs da Robocup, cada uma foi agendada de acordo com a disponibilidade dos participantes.

Para avaliar a métrica M1, o código produzido foi analisado. O RoboSocBG gerou 185 linhas de código (LoC) para o novo comportamento. Esses LoCs foram suficientes para executar o novo comportamento na ferramenta SPlanner. Nenhum LoC extra foi inserido. Portanto, para a ferramenta SPlanner, a métrica M1 resulta em 100% dos códigos gerados.

Usamos os códigos da equipe BahiaRT para testar os comportamentos produzidos no estudo de caso. A integração exigiu a inserção de 222 LoCs no código-fonte do time BahiaRT. Assim, para o teste de execução do FSF com o time contendo o novo comportamento gerado, a métrica M1 atingiu a taxa de completude de 45%. As LoCs inseridas não foram desenvolvidas pelo pesquisador, mas substituídas por códigos já existentes. A equipe executou com sucesso o setplay criado com o novo comportamento, garantindo a validação semântica dos códigos gerados pelo RoboSocBG (Métrica M2).

Todos os participantes concordaram com o aumento da produtividade na criação de comportamentos com o RoboSocBG, quando comparado com a codificação manual.

5.3. Ameaças à validação

Para que não houvesse diferença no tratamento da quantidade de linhas de código geradas em cada replicação do estudo, todos os participantes definiram o mesmo comportamento; convidamos apenas participantes com alguma experiência em futebol de robôs, para que houvesse familiaridade com os conceitos relacionados a esse domínio de aplicação. Para a validade externa, foram selecionados apenas participantes que estão atuando na Robocup. Para que houvesse uma diversidade de amostras, tivemos a participação de integrantes de duas equipes distintas. Contudo, realizar o estudo com um número maior de participantes é importante para validar e generalizar os resultados obtidos. Finalmente, considerando as ameaças à confiança, relacionadas ao nível de dependência dos dados e do estudo ao pesquisador, o material utilizado no estudo foi disponibilizado eletronicamente em https://github.com/Raunis/RoboSocBG.

6. Conclusão

Essa dissertação propôs uma solução DDM que apoia a criação de novos comportamentos atômicos para definição de *setplays* em futebol de robôs. Atualmente, o desenvolvimento de novos comportamentos é realizado de forma manual, uma prática que tem se mostrado complexa, improdutiva e sujeita a erros.

A solução RoboSocBG fornece uma linguagem específica de domínio para definir comportamentos independentes de plataforma e transformações para gerar código. As ferramentas SPlanner e FSF, com grande aceitação pela comunidade, foram adotadas como plataformas de destino. Outras plataformas serão consideradas em breve.

O RoboSocBG foi validado nos testes em nosso laboratório e no estudo de caso com times da subliga de simulação 3D da RoboCup. Os resultados evidenciam a aplicabilidade da solução e demonstram que ela não requer conhecimento prévio da linguagem de programação adotada nas plataformas alvo para sua utilização.

Estamos trabalhando na replicação de estudos de caso com novos comportamentos e planejando replicações com participantes sem contato prévio com o FSF ou SPlanner. Nosso objetivo é medir o tempo gasto criando novos comportamentos utilizando o RoboSocBG em comparação com o tempo gasto treinando desenvolvedores para realizar essa tarefa manualmente.

A pesquisa de futebol de robôs pode ser replicada para diferentes contextos de SMAs. Nosso laboratório está reutilizando a experiência do futebol robótico para trabalhar com drones autônomos e planejamos usar o RoboSocBG para definir o comportamento do drone e gerar código para plataformas de simulação.

Referências

Andrade, J. P. B. et al. (2016). Uma abordagem com sistemas multiagentes para controle autonomo de casas inteligentes.

Brambilla, M., Cabot, J., and Wimmer, M. (2012). Model-driven software engineering in practice.

Cravo, J., Almeida, F., Abreu, P. H., Reis, L. P., Lau, N., and Mota, L. (2014). Strategy planner: Graphical definition of soccer set-plays. *Data & Knowledge Engineering*, 94:110–131.

Cravo, J. G. B. (2011). Splanner - uma aplicação gráfica de definição flexível de jogadas estudadas no robocup. *Master degree dessertation in Informatics and Computing Engineering*. https://repositorio-aberto.up.pt/bitstream/10216/62120/1/000149781.pdf Acesso em: 15 de novembro de 2021.

Hogg, T. and Sretavan, D. W. (2005). Controlling tiny multi-scale robots for nerve repair.

INFORMATIK (2011). Simrobot. http://www.informatik.uni-bremen.de/simrobot/index_e.htm Accessed 15 November 2022.

Jung, C. et al. (2019). Computação embarcada: Projeto e implementação de veículos autônomos inteligentes.

Konolige, K. and Nilsson, N. J. (1980). Multiple-agent planning systems. In *Proceedings of the First AAAI Conference on Artificial Intelligence*, AAAI'80, page 138–142. AAAI Press.

Magalhães Mascarenhas, A., Maciel, R., and Andrade, A. (2015). Towards a metamodel design methodology: Experiences from a model transformation metamodel design. *International Conference on Software Engineering and Knowledge Engineering*.

Marques, F. T. (2010). Generic coordination methodologies applied to the robocup simulation leagues. *Master degree dessertation in Informatics and Computing Engineering*. https://repositorio-aberto.up.pt/bitstream/10216/63353/1/000147444.pdf. Accessed 15 November 2022.

Meessen, K. (2011). Turtle simulator. http://www.techunited.nl/wiki/index.php?title=Simulator Acesso em: 15 dez. 2021.

Mota, L. et al. (2015). Collaborative behavior in soccer: The setplay free software framework. *Lecture Notes in Computer Science*, 8992:709–716.

OMG (2008). Mof model to text transformation language, v1.0. *Object Management Group*. https://www.omg.org/spec/MOFM2T/1.0/PDF. Accessed 24 October 2022.

Paraschos, A., Spanoudakis, N. I., and Lagoudakis, M. G. (2012). Model-driven behavior specification for robotic teams. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, page 171–178, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Pessin, G. et al. (2007). Simulação virtual de agentes autônomos para a identificação e controle de incêndios em reservas naturais.

Reis, L. P. et al. (2010). Playmaker: Graphical definition of formations and setplays. In *5th Iberian Conference on Information Systems and Technologies*, pages 1–6. https://ieeexplore.ieee.org/document/5556598.

Robocup. Objective. *www.robocup.org/objective. Accessed 23 October 2022*.

Russell, S. J. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, Berkeley.

Solingen, R., Basili, V., and Caldiera, G.and Rombach, H. (2002). *Goal Question Metric (GQM) Approach*. John Wiley & Sons, Hoboken.