One of the research topics on multi-agent systems focuses on the development of mechanisms such as plans to empower a team of agents to cooperate in order to perform complex tasks. In many cases, the definition of these plans are based on a specific and rather complex grammar and stored in structured text files.

In the context of the 2D simulated Robotic Soccer domain, a set-play language was proposed to coordinate the execution of teammates’ behaviors to improve a team’s overall performance. The process of manually writing set-play definitions is hazardous and can benefit from the use of a graphical tool to reach new users and allow typical users to become more productive.

This work presents such a tool for which several experiments were run to measure its usability with forty two users by having them perform a set of tasks for which their execution time, number of errors and satisfaction were recorded.

The tool reduced the previous average time required to completely define a set-play by 90% and enabled even non-expert users to use it. Moreover, users were on average satisfied with SPPlanner having ranked it with a score of 77 (out of 100) using a System Usability Scale questionnaire.

1. Introduction

Artificial Intelligence and Robotics have been two areas of research which have received a great deal of attention over the past few years.

These areas of research have been fostered particularly by international initiatives like RoboCup which accommodates many challenging competitions. From these competitions the one with the most fans is undoubtedly the soccer competition due to its wide acceptance over the world. This competition places two teams of robotic agents up against each other to dispute victory in a soccer match. Teams have been improving performance by creating new strategies that currently consider the definition of strategic positioning [1–6] based on formations, tactics and set-plays [7–9].

A set-play can be part of a team’s strategy and is a widely known concept in real soccer as well as in other cooperative sports to leverage a competitive advantage against an opposing team. A set-play can be described as a structured plan that describes courses of actions that a subset of players in a team should take based on the current state of a game. Some attempts to make use of set-
plays have already been made in the robotic soccer domain, however the knowledge for their definition and execution is tightly coupled (hard-coded) with the soccer player agent internal implementation. A framework that promotes the decoupling of the knowledge of set-plays from the soccer player agents internal implementation using a s-expression language has recently been developed [9]. However, writing set-play definitions manually is a harsh, error-prone and time consuming process. For these reasons, a graphical tool, named Strategy Planner (SPlanner), is proposed to speed-up the definition of set-plays and reduce the amount of errors committed by abstracting the complexity of the grammar from the end users.

The rest of this article is organized in the following manner. Section 2 describes some of the related work done in the context of strategy definition, with a particular emphasis on the definition of set-plays. Section 3 describes the functionality and some usage examples of the Set-play framework used as the basis of this work. Section 4 presents the developed graphical user interface (GUI) of the SPlanner tool, focusing on its integration with the Set-play framework. Section 6 describes the methodology used to perform experiments in order to validate the usefulness of the developed tool. Section 7 presents an analysis of the results obtained from the experiments performed to assess the usefulness of SPlanner. Section 8 draws the main conclusions from the developed work and establishes some pointers for future work.

2. Related work

The general concept of strategy can be described as a previously planned and typically complex behavior whose goal is to make use of available resources in the most efficient and effective way [10]. The concept of strategy has been widely adopted in several domains and its definition has evolved to match the specificities of each domain. In collective sports, particularly in the soccer domain, the concept of strategy has been the main driver for the improvement of the game quality of teams over the years.

The use of technologies plays an important role in the improvement of strategies. They allow for games to be recorded and afterwards analyzed. Using tailored softwares (e.g. simulators, tactical panels, analyzer) the specification of strategies and their communication to the interested parties (coaches and players) is eased.

In the human soccer domain, strategy is considered to be a key point for the difference between two teams. Concepts such as game rules (how the game must be played), player roles (what type of behavior is expected from a player), coach instructions (advice given to players to adjust their behavior to become more adequate in a match), formations (players positioning in the field) and set-plays (predefined plans of action used to gain advantage over an opponent) can all play a part in a team’s strategy.

In the robotic soccer domain, particularly in the RoboCup competition, the previous concepts have already been researched and experimented in different teams [11] of different leagues mainly with the goal to build better coordination mechanisms that will allow teams to improve their performance and have an advantage over their opponents. Some exemplar instances of the implementation of such strategic concepts include: general dynamic positioning [1–4], coaching [12,13], defensive positioning [6], set-plays [9,7,8] and offensive positioning [5]. As corroborated by the previous citations, these concepts have been mostly tested in the 2D soccer simulation league, in particular because it provides a standardized and robust platform around which several community open-source tools have been developed that ease the development of new ideas.

Several tools have been developed with the goal of assisting the definition of strategies in real soccer (e.g. Coach-Helper, Academy Soccer Coach, ForCoach Tactics, ForCoach of Soccer, Tactics Manager). These software applications are commercial and its functionality is mainly focused on the use of soccer tactical panels for the definition of team formations. In order to assess the usability of the tools some user tests and a heuristic evaluation [14] based on the following set of heuristics [15] was conducted:

- Ensure the visibility of system status;
- Adequately match system to real world concepts whenever possible;
- Provide the user control and freedom;
- Make use of standards and be consistent;
- Prevent user errors;
- Favor recognition rather than recall when using dialogs;
- Provide flexibility and efficiency of use;
- Develop esthetic and minimalist designs for dialogs;
- Help users recognize, diagnose, and recover from errors;
- Provide concise and step-oriented documentation focused on user tasks.

The results of the heuristic evaluation led to several conclusions, from which the most important will be highlighted. The tool that was easier to use was ForCoach Soccer which when combined with ForCoach Tactics became more suitable for defining team’s tactics and formations. Although these tools use the set-play concept, none of them allow to graphically or formal define set-plays or even store the set-plays created.

The video-games industry has registered a trend of growth over the last few years, mostly due to the high level of realism that creators were able to imprint in their games. In this industry, there are many examples of soccer games (e.g. Hattrick, Online Football Manager, Virtual Manager, FIFA Soccer 2011, Pro Evolution Soccer 2011, Championship Manager 2010, Football Manager 2011, FIFA Manager 2011) made available to users in different platforms (e.g. online web-based, game consoles, PC) that empower them to define some kind of strategies. Once again, the video-games reality presented the same drawbacks exposed in the human soccer scenario. The online games that use web browsers are too simplistic and offer few tools for the definition of strategy. The soccer
simulators tend to offer a more appealing set of tools for its diversified end users in order to allow them to adjust the complexity to best suit their skills and explore different playing styles. From the previous examples of soccer games, only FIFA Soccer 2011 and Pro Evolution Soccer 2011 provided some tools to change a team’s strategy during a game although with different levels of detail. In particular, FIFA Soccer 2011 allows the definition of positions of players in the field and their mentalities. On the other hand, Pro Evolution Soccer 2011 allows the selection of predefined game strategies for different states (e.g. times) of the game. Team management games are the ones which present the most rich set of functionality devoted to the definition of strategies, since this is their main focus. Regrettably, Championship Manager 2010 was the only software to provide a worthy tool to build set-plays and was an inspiration for this work.

The robotic soccer community has also developed several tools to enable the definition of strategies for teams such as Playmaker [16], Team Designer, Matchflow, B-Smart Strategist [17] and Formation Editor [18]. Playmaker constitutes a previous attempt to build a graphical tool for defining set-plays [9] but the resulting GUI was not suitable to be used by non-expert users because it was not intuitive.

From the survey conducted, it was concluded that the most complete and suitable tools for defining strategies for soccer teams are embedded in soccer video-games. In the robotic soccer domain, only a few teams have reached the goal of using high-level multi-agent strategies and thus the few existing tools are very specific for each team.

3. Set-play framework

The set-play framework [9] provides a language specification for defining plans (set-plays) for the soccer domain, a built-in parser and an engine that allows them to be interpreted and executed at run-time.

Set-plays can be particularly useful in some situations (better exploit empty spaces in the opponent’s goal area) to help the team achieve a competitive edge. Moreover, the continuous improvement of soccer agents tactics and skills requires the development of new strategies to counter them and thus the definition of set-plays can prove useful for this purpose. Set-plays can also be a means to create mixed teams (composed by heterogeneous players) because players only have to follow the steps in the set-play to be able cooperate. A set of defined set-plays can be reused in different soccer games and integrated with other existent team strategy mechanisms such as tactics and formations to better cope with opponents.

This framework has been tested in the soccer simulation leagues (2D and 3D) and the Middle Size leagues of the RoboCup competition [19]. Besides the Simulation 2D league, the framework has been tested in the soccer 3D simulation leagues (also in the context of FC Portugal team) and the Middle Size league (in the context of CAMBADA team) of the RoboCup international competition [19–21]. These are very distinct leagues from the simulation 2D. The simulation 3D league uses teams of 11 simulated humanoid NAO robots. The simulator is very realistic and thus, teams must control all robot motors in order to be able to make them walk, getup or kick the ball. In the middle-size league teams of 5 real autonomous robots play soccer in a 18 × 12 m field using a standard FIFA ball. Robots height is limited to 80 cm, weight is limited at 40 kg and the robots horizontal shadow must fit inside a square of 50 cm. Robots are completely autonomous, although they are allowed to communicate each other, and all sensors must be mounted on the robots. The framework helped the two teams to achieve very good results in both leagues in the last three years. FC Portugal 3D team was the champion of the last three European RoboCup championships (2012–2014) and achieved several other awards such as third place in RoboCup world championship 2013 and two scientific award challenges at RoboCup. CAMBADA team achieved also very good results including third place in the last three RoboCup world championships (2012-2014) and two scientific challenges.

3.1. Set-play formal definition

Setplays are meant to be freely definable through text files that can be read upon agent launching. To accomplish this goal, Setplays must have a defined syntax on which to base both the writing of Setplays and the parser which will load them. In order to keep this syntax familiar to the RoboCup community, it was based on s-expressions [22], which are already the building blocks of the communication language with the server and with the coaching language, both from the Soccer Simulation 2D league.

The syntax of this Setplay definition language, written as a set of BNF statements [23], is included bellow. In terms of BNF syntax, terminal symbols are in regular type, non terminals in italics between the ‘<‘ and ‘>’ symbols, and BNF specific symbols in bold, with the following definitions were used:

::=   definition;
-    alternative;
?    0 or 1 occurrences of the previous term;
*    any amount of occurrences of the previous term;
+    1 or more occurrences of the previous term.

Some comments will be inserted amidst the text to clarify some options. Generally speaking, the language tried to use the concepts already present in CLang [24], the standard coach language in the 2D Simulation league. In some cases, though, practical experience showed that additional concepts were necessary, which were added to the language.
In general terms, one should emphasize the following options:

- All references to players are done through the type `PLAYER_REFERENCE`;
- All operators (`+`, `-`, `*`, `/`, `==`, `!=`, `b`, `b=`, `N`, `N=`) are prefixed;
- Arguments of objects and functions have added labels, prefixed by a colon (e.g. `:label`);
- All lists are built through the s-expression with the functor `list`, like, e.g., `(list <args >)`.

At general level, one will consider a Setplay a conjunction of a `Parameter` list, a `Player Reference` list, an `Abort Condition` and a `Step` list.

\[
\text{<SETPLAY> ::= (setplay <PARAMETER_DEFINITION_LIST>?}
\text{<PLAYER_REFERENCE_DEFINITION_LIST>}
\text{:abortCond <CONDITION> <STEP_LIST>)}
\]

Parameters will be characterized by their name and type. The available types are only numbers and spatial entities (points and regions).

\[
\text{<PARAMETER_DEFINITION_LIST> ::=}
\text{ :parameters (list <PARAMETER_DEFINITION>+)}
\text{ <PARAMETER_DEFINITION> ::= (parameter :name <VAR> :type}
\text{ <TYPE>)}
\text{ <TYPE> ::= integer | decimal | region | point}
\]

Players can be referred to in two different ways: either through a full identification by team and jersey number, or through the name of the role they will play in the Setplay. When in the scope of the parameter definition, the full syntax of player roles and player identification must be employed, as defined in `PLAYER_REFERENCE_DEFINITION`. In other situations, inside the Setplay definition, player roles can simply be referred to by their names, as per `PLAYER_REFERENCE` definition.

\[
\text{<PLAYER_REFERENCE_DEFINITION_LIST> ::=}
\text{ :players (list <PLAYER_REFERENCE_DEFINITION>+)}
\text{ <PLAYER_REFERENCE_DEFINITION> ::= (playerRole :name <VAR>)}
\text{ | (player :team <TEAM> :number <PLAYER_NUM>)}
\text{ <TEAM> ::= our | opp}
\text{ <PLAYER_NUM> ::= [0-9] | 10 | 11}
\text{ <PLAYER_REFERENCE_LIST> ::= (list <PLAYER_REFERENCE>+ )}
\text{ <PLAYER_REFERENCE> ::= <VAR>}
\text{ | (player :team <TEAM> :number <PLAYER_NUM>)}
\]

Steps are, as said before, the main building blocks of Setplays, where they represent intermediary stages in Setplay execution. As such, Steps must be precisely characterized. Their id allows Steps to be referred in Transitions from other Steps. `WaitTime` represents the time that needs to elapse before trying to transition to another Step, while `abortTime` is the time after which the Setplay is to be aborted if no transition to another Step is possible. Both time intervals must be non-negative integers. The `Condition` can be seen as a pre-condition that must be satisfied before entering the Step. The Step also contains a list of Participants and Transitions, described hereafter.

\[
\text{<STEP_LIST> ::= (list <STEP>+ )}
\text{ <STEP> ::= (step :id <NON_NEG_INTEGER>}
\text{ :waitTime <NON_NEG_INTEGER> :abortTime <NON_NEG_INTEGER>}
\text{ :participants <PARTICIPATION_LIST> :condition <CONDITION>}
\text{ :transitions <TRANSITION_LIST>)}
\]
A Participation identifies a player that takes part in the Step, and may optionally also determine the player’s initially desired location for this Step, which shall be done through the usage of the ‘at’ functor.

```xml
<PARTICIPATION_LIST>::= (list <PARTICIPATION>+)
<PARTICIPATION>::= <PLAYER_REFERENCE>
 | (at <PLAYER_REFERENCE> <REGION>)
```

Transitions are the options to move from one Setplay Step to another (through nextStep), or to finish or abandon Setplays. In the case of transitions to other Steps, a list of Directives can be used to determine the actions the player should execute, in the case of the Do directive, or avoid to execute, in the case of Dont.

```xml
<TRANSITION_LIST>::= (list <TRANSITION>+) | <TRANSITION>
<TRANSITION>::= <NEXT_STEP> | <FINISH> | <ABORT>
<NEXT_STEP>::= (nextStep :nextStepNumber <NON_NEG_INTEGER>
 :condition <CONDITION> :directives <DIRECTIVE_LIST>)
<FINISH>::= (finish :condition <CONDITION>)
<ABORT>::= (abort :condition <CONDITION>)
<DIRECTIVE>::= (<DIRECTIVE_NAME> :players
 :actions <ACTION_LIST>)
<DIRECTIVE_NAME>::= do | dont
<DIRECTIVE_LIST>::= (list <DIRECTIVE>+)
```

Actions, in the scope of the Setplay Framework, represent abstract concepts that model the different high-level behaviors a player might need to execute. To keep the maximum compatibility with previously existing concepts, all the actions defined in CLang [24] were imported, with similar meanings and syntax. Some new actions had to be added such as:

- **receiveBall** receive a pass from another player, which is, implicitly, the ball owner. Takes no arguments.
- **attentionTo** (object or region) direct visual and/or auditive attention to a particular object or region, taken as argument.
- **seq** action sequence, is simply an aggregator of actions, that are given as arguments, and should be executed in sequence.
- **markGoal** position in a location suitable for avoiding opponent shots to enter the goal.
- **moveToOffSideLine** move near the offside line, avoiding at the same time to cross it. The argument indicates the y-coordinate of the desired location.
- **stop** stop as quick as possible, making sure that any remaining inertia is adequately counter-balanced.

```xml
<ACTION>::= (pos :region <REGION>)
 | (bto :region <REGION>)
 | (bto :players <PLAYER_REFERENCE_LIST>)
 | (mark :players <PLAYER_REFERENCE_LIST>)
 | (markl :players <PLAYER_REFERENCE_LIST>)
 | (markl :region <REGION>)
 | (markGoal) | (oline :region <REGION>)
 | (clear :region <REGION>) | (hold)
 | (dribble :region <REGION>) | (shoot)
 | (tackle :players <PLAYER_REFERENCE_LIST>) | (intercept)
 | (receiveBall) | (stop) | (attentionTo :region <REGION>)
 | (attentionTo :object <OBJECT_LIST>)
 | (moveToOffSideLine :y <DECIMAL>) | (seq <ACTION>+)
<ACTION_LIST>::= (list <ACTION>+) | <ACTION>
```
Objects are further characterized as being mobile or static, and can include all the agents (human and robotic) as well as passive objects that can be possibly on the pitch. Static objects are not yet exhaustively defined.

```
<OBJECT_LIST>::= (list <OBJECT>+ ) | <OBJECT>
<brAND_OBJECT>::= <STATIC_OBJECT> | <MOBILE_OBJECT>
<brAND мобильיכול OBJECT>::= <POST> | <FLAG> | ...
<brAND MOBILE_OBJECT>::= (ball) | <PLAYER_REFERENCE> | (referee)
```

Conditions were originally based on CLang [24]. In this case, though, to better tune the some specific situations, several new Conditions had to be created, as follows:

- `canPassPl` check if some player in the `from` list can execute a pass to some player in the `to` list.
- `canPassReg` check if some player in the `from` list can execute a pass to the region given as the `to` argument.
- `canShoot` check if some player in the `players` list can shoot at goal.
- `nearOffsideLine` check if some player in the `players` list is near the offside line, for situations where a forward behind this line is to be done.

Conditions regarding score and time are done through dedicated keywords and the usual comparison operators.

```
<CONDITION>::= (true) | (false)
  | (pos : players <PLAYER_REFERENCE_LIST> : min <INTEGER>
    : max <INTEGER> : region <REGION>)
  | (pos : region <REGION>)
  | (bonner : players <PLAYER_REFERENCE_LIST>)
  | (playm <PLAY_MODE>)
  | (canShoot : players <PLAYER_REFERENCE_LIST>)
  | (canPassPl : from <PLAYER_REFERENCE_LIST>
    : to <PLAYER_REFERENCE_LIST>)
  | (canPassReg : from <PLAYER_REFERENCE_LIST>
    : to <REGION>)
  | (nearOffsideline : players <PLAYER_REFERENCE_LIST>)
  | (and <CONDITION_LIST>)
  | (or <CONDITION_LIST>)
  | (not <CONDITION>)
  | (COND_COMP)
<brAND CONDITION_COMP>::= <TIME_COMP>
  | <OPP_GOAL_COMP>
  | <OUR_GOAL_COMP>
  | <GOAL_DIFF_COMP>
<brAND TIME_COMP>::= (time <COMP> <INTEGER>)
<brAND OPP_GOAL_COMP>::= (opp_goals <COMP> <INTEGER>)
<brAND OUR_GOAL_COMP>::= (our_goals <COMP> <INTEGER>)
<brAND GOAL_DIFF_COMP>::= (goal_diff <COMP> <INTEGER>)
<brAND COMP>::= < | <= | == | != | >= | >
<brAND PLAY_MODE>::= bko | time_over | play_on | ko_our | ko_opp
  | ki_our | ki_opp | fk_our | fk_opp | ifk_our
  | ifk_opp | ck_our | ck_opp | gk_opp | gk_our
  | gc_our | gc_opp | ag_opp | ag_opp
<brAND CONDITION_LIST>::= (list <CONDITION>+ )
```

Regions model sections of the pitch, and are organized around the usual concepts of arcs, triangles and rectangles. In order to use names commonly used for well known regions, a new region type was added (Named Region), which may have a wide range of pre-defined values, inspired in part by the ones defined by Coach-Unilang [13].
As for points, there is a wide number of different operators to refer to static and dynamic points, as well as to perform some operations, namely translations, on these.

3.2. Execution of a set-play

Setplays are flexible plans that, like in real soccer, may be instantiated in real-time to face distinct opponents. In order to execute set-plays, all players keep a play book with all of the set-plays definitions to be used in a match. The state of the match is continuously monitored by the set-play engine to assess whether the execution of a set-play should be started or stopped. A set-play is considered for execution whenever the entry Condition of its first Step becomes true and no other set-play is currently being executed. Whenever there are multiple eligible set-plays for execution a plugin algorithm is used to select the most relevant. Thus, the set-plays to be used are selected during the game according to the opponent and game situation. For example the playbook may include distinct corner-kick setplays to be used against different opponents and distinct corner-kick setplays to be used depending on the game situation (that includes, among other factors the current score and time). The decision is up to the coach/programmer that has the advantage of being able to quickly and graphically define new setplays to face a given opponent, without the need to change or recompile the teams code. After a set-play is chosen for execution a plugin algorithm (e.g. choose players nearest the Player References expected locations in the first Step) is run by all players concurrently to determine the most suitable allocation of concrete players to the Player References in the Set-play definition. The player that determines himself to be the Leader of the first Step of the Set-play will inform other Set-play participants of the start of the new Set-play and the role that each will play. If more than one player determines himself to be the Leader of a new Set-play, for now the player with the lowest jersey number will be deemed the effective Leader.1. In each Step the Leader instructs other Set-play participants of which transition to follow in order to carry on with the execution of the set-play. All participants continuously monitor the execution of the Set-play to decide whether to continue its execution. The entire execution of a Set-play will be ended if any of the following situations occur:

- The Abort condition of the Set-play is evaluated as true;
- The execution of the current Step has not been started, the value of Wait Timer for the current Step has been reached, the entry Condition of the Step is defined and evaluates to false;

\[\text{<REGION> ::= <VAR> | (regVar : name <CLANG_STR> )}\\ | \text{(regVar : name <CLANG_STR> : value <REGION>)}\\ \text{| <POINT> | (regnamed : name <REGION_NAME>)}\\ \text{| (arc : center <POINT> : radius_small <REAL> : radius_large <REAL> : angle_begin <REAL> : angle_span <REAL>)}\\ \text{| (triang : points (list <POINT> <POINT> <POINT>))}\\ \text{| (rec : points (list <POINT> <POINT>))}\\\]

\[\text{<REGION_NAME> ::= field | outside | our_middle_field | their_middle_field | left | right | far_left | mid_left | centre_left | centre_right | mid_right | far_right | our_back | their_middle | their_front | their_back | s1.1 | s1.2 | s1.3 | s1.4 | s1.5 | s1.6 | s1.7 | s1.8 | s1.9 | sr.1 | sr.2 | sr.3 | sr.4 | sr.5 | sr.6 | sr.7 | sr.8 | sr.9 | our_penalty_box | our_goalie_area | their_penalty_box | their_goalie_area}\\\]

\[\text{<POINT> ::= <VAR> | (ptVar : name <CLANG_STR> )}\\ | \text{(pt : x <REAL> : y <REAL>) | (pt ball)}\\ \text{| (pt : player <PLAYER_REFERENCE>)}\\ \text{| (ptRel : x <REAL> : y <REAL> : pt <POINT>) | ( <POINT_ARITH> )}\\\]

\[\text{<POINT_ARITH> ::= (<OP> <POINT> <POINT>)}\\ \text{<OP> ::= + | - | * | /}\\\]

\[\text{<CLANG_STR> ::= [0-9A-Za-z().+-*/?<>\_]*}\\\]

\[\text{<VAR> ::= [A-Z][a-zA-Z0-9]*}\\\]
• The execution of the current Transition has not been started, the value of Abort Timer for the current Transition has been reached, the entry Condition of the Transition is defined and evaluates to false.

Setplays also consider the opponent players during their execution in the game. They include conditions such as canPassPl (can pass to player), canShoot (can shoot to goal), canPassReg (can pass to region), pPos (player position). These conditions enable to consider the teammates and opponents in an appropriate way since their logic value depends on the concrete positions of the teammates and opponents during the setplay execution. In fact, the dependence of the setplay execution on the opponent players is even greater since setplays may include branches. For each possible transition the setplay framework uses the agent specific evaluation of each of the possible actions and selects, from the possible actions, the one that achieves higher evaluation. Thus, not only the transition to the next step but the concrete step and actions selected depend on both teammate and opponents behavior and position during the setplay execution.

4. Graphical definition of set-plays

The SPlanner tool was developed in C++ for the Linux platform and makes use of Qt1 graphical libraries to ease the integration with the majority of tools produced by the RoboCup Soccer community. An overview of the SPlanner tool architecture is presented in Fig. 1.

The development of the tool was carried out with modularity in mind, in order to easily allow the future integration of new strategy modules (e.g. formation editor, tactics manager). Several Application Programming Interfaces (APIs) interfaces were defined to abstract the import and export functionality of other plan-based frameworks, external programs and game log viewer. The application allows an integrated test and debugging of set-plays using the Soccer Server [25] simulator and monitor and the FC Portugal Visual Debugger [26], the latter two make use of generated game logs.2

4.1. Interface design

The interaction with the main interface of SPlanner, presented in Fig. 2, is done primarily using the mouse and keyboard.

This interface depicted in Fig. 2 is organized in three main areas: the menu bar, the set-plays separators (for editing more than one set-play at once) and the set-play workspace. In the menu bar, the user can find several actions grouped in three sub-menus: File, Setplay and Help. The File menu contains options to create, import and export a set-play and exit the application. The Setplay menu contains options to start a test of a devised set-play, debug a previously executed set-play and to configure the location of the binaries of the Soccer Server simulator, Monitor tools and the FC Portugal Visual Debugger as well as the team binaries to use in the test. The Help menu contains documentation that helps to better understand what are the tasks that can be executed with the tool and how they are supposed to be executed.

1 Qt is a cross-platform application and UI framework which is available at http://qt.nokia.com/products/.
2 Structured representation of soccer scenes (players and ball positions) and play-modes.
The central area of the set-play workspace is essentially a metaphor for the soccer pitch divided in two distinguished areas consisting of its inner bounds (game area) in which the players participating in the set-play will be positioned and their respective actions defined and the outer bounds in which a substitution bench highlighted in brown will contain the players not participating in the set-play. In the left area a panel was defined that holds general information that describes the set-play and its abort conditions (step-independent), the flow of execution of a set-play mapped into a graph (further explained in Section 4.2) and the abort and wait times for the currently chosen step. In the bottom area of this workspace a collapsible panel was create to provide additional relevant information regarding the current active player. Just above this last area, the absolute position (coordinates) of the mouse cursor (the same as the players when being dragged) is provided in real-time to the user.

A wizard was designed to assist the user in the creation of set-plays in a more intuitive fashion. Based on the experience gathered from the analysis of similar software tools and after studying the typical task flow for the creation of a set-play the following steps were defined:

1. Define the type of set-play (offensive or defensive) to create;
2. Choose the game situation (e.g. free-kick) that triggers the set-play;
3. Select the location in the field (e.g. opponent front wing) that combined with the game situation will trigger the set-play. It should be noted that some positions will be omitted based on the chosen game situation (e.g. kick-off) as they can only start from one specific location.

If a location in the field is chosen in third step of the previous procedure it will be visually highlighted with a textured shape for the user to know where the set-play will be triggered. Moreover, if a region in the field is chosen, the tool will constrain the ball position and consequently the leader, which is assumed to own the ball in every step. Although, the set-play grammar allows a leader to be any arbitrary player participating in a step, in the majority of cases in real or simulated soccer he will be the player that owns the ball in offensive set-plays. This assumption makes no sense for defensive set-plays as no teammate owns the ball at its start.

The players that will be participating in the set-play must all be added in the first step of the set-play, although it is not mandatory that they participate in all subsequent steps. Afterwards, the user can only define new player positions for subsequent steps implicitly by assigning actions to them in transitions that lead to those steps. The tool applies an action effector model\(^3\) to the players and the ball to estimate their future positions in subsequent steps of the set-play which cannot be changed.

In order to assign a player to the set of participants of a set-play, the user just needs to drag him from the team's substitution bench into the field area. At anytime during the design of a step of a set-play, it is possible to include a non-participating player (he will have

\(^3\) This model can be parametrized to describe the constraints of the application domain.
a white translucent jersey shown at the right in Fig. 3) to that step, but only if he has already been added as a participant in the first step of the set-play. This can be accomplished automatically by adding an action to this player. The execution of the inverse action, dragging a player to his team’s bench or simply out of the field, removes him from the list of the step participants. This metaphor will appear natural for users with some knowledge of the soccer domain as a strong connection exists with the reality in which players that are sitting on the substitution bench are inactive and thus not participating in the game, contrarily to the players that are positioned within the field boundaries.

Players are represented by a numbered and colored jersey that provides information to about their status (see Fig. 3) and can be selected or dragged. Whenever a user clicks on a player’s jersey that player becomes active and his jersey becomes yellow. An active participant is a player for which actions can be defined in the context of a particular step. In some set-plays it might not be required that a participant does something in every step. To identify this status on a player a white translucent jersey is used, as previously mentioned. These players’ statuses will be made visible whenever the user attempts to redesign a step which does not involve all participants of the set-play. This representation is relevant for the user as at any given step it provides him with visual information that allows him to see where every step participant would be based on their previously executed actions. The lead player is identified by a white jersey with a ball at the top right corner.

A right mouse-click on a step participant triggers the context menu depicted in Fig. 4 from which the user can specify its actions, initial position or simply remove him from the step. In this context menu, only feasible actions will be presented to the user (e.g. a pass action will only be present if the player owns the ball). Furthermore, for some specific feasible actions presented to the user, there might be only small set of relevant options to choose from and these are thus filtered for the user. For instance, in a step scene with three players A, B and C if a pass action is initially chosen for A the only possibilities for completing its definition is to select player B or C as the receivers and so only this information is presented to the user.

### 4.2. Execution flow of a set-play

In order to provide a general comprehension of the defined set-plays execution flows a graph visualization was designed. This graph contains a representation of identifiable steps (numbered circles) and the defined transitions between them (directed arrows) in the set-play, allowing the user to understand all its possible execution flows as depicted in Fig. 5.

There can only exist one start step and any arbitrary number of intermediate and final steps.

To facilitate the distinction between different types of steps (start, intermediate or final) three different color schemes were applied when rendering the drawing. The start step is filled in black and has the number 0 and the intermediate and final steps are filled in gray. Moreover, the final step is distinguished from an intermediate step by adding a double black contour to it. The rendering of the graph is done automatically in real-time as the set-play definition is updated. This rendering process was designed to be legible by clearly organizing its elements separating in horizontal levels with equally distributed spacing between step circles and alignment between different levels of execution.
A new transition is added by selecting a step in the graph by right-clicking it and choosing its destination to an existing step or a new step as intended. Currently, transitions between two existing steps are not supported at this time, although they are possible. In order to achieve this the system needs to reconcile the effects of executing the actions specified for the transition from the source step to the target step which has initial constraints for the positionings of its participants. Thus, the system will need to infer which actions (essentially positionings) need to be executed by each participant in order to preserve the conformity with the participants’ positionings at the beginning of the target step.

4.3. Defining actions for participants

SPlanner has currently built-in support for eight actions, for which different iconographies were defined (see Fig. 6) to make them clearly distinguishable for the users designing a set-play.

From these actions, five require the step participant to own the ball at the time of its execution (direct pass, forward pass, dribble, hold the ball and shoot) contrarily to the remaining three (wait, run and position near the opponent offside line). The actions presented to the user when he selects a participant are filtered based on this criteria to prevent semantical errors (e.g. defining a dribble action for a participant that does not own the ball).

4.4. Positions of players and action targets

The positions of the set-play participants in the start of a step, as well as their predicted target positions inferred from the effects of their associated actions, can be undefined, absolute (e.g. field coordinates) or relative (e.g. to the ball or participant). Each position type is distinguished visually in the set-play representation using the iconography shown in Fig. 7.

Player and action target positions are assumed to be absolute by default as the most common scenarios for the definition of a set-play make use of well defined locations. These absolute positions have no additional visual element apart from the iconographies chosen to represent the players and their actions.

Undefined positions are used to add flexibility to the execution of a set-play (e.g. prevent players from running back to a location worse than the currently occupied), as they do not impose a strict position for a step participant (e.g. the leader) at the start of a step. The uncertainty of the execution of actions in a transition between steps can make the participants occupy a position different than the one that might be expected, but that could nonetheless be relevant to proceed with the execution of the set-play.
The letter R is placed on the top left corner of a player’s jersey and on the middle of the iconography of an action to represent an initial relative position towards a player and for the intended action respectively. This letter can be followed by a number identifying the player to whom that player is relatively positioned (e.g. R7) or the letter B if he is positioned relatively to the ball. When the letter D is placed on the top left corner of a player’s jersey it represents an undefined position for that player.

In order to demonstrate the simplicity allowed by SPLanner for defining a set-play, a step-by-step example of the process for defining a corner-kick set-play with three participants is described in Fig. 8 and its manual definition is presented in the Appendix A.

5. Promoting the usability of the tool

Several precautions were taken to promote a good usability of the tool. The process of designing the GUI was guided by the set of well known heuristics [15] described in Section 2.

To avoid unnecessary interactions for expert-users some keyboard accelerators were defined in the tool to speed-up the definition of actions for the selected player. For instance, if the user presses the letter P in the workspace when a player that owns the ball is selected, a pass action is initiated from that player to the coordinate where the mouse pointer is located.

Also, valid names for the set-play and roles for the players are automatically defined upon the creation of a set-play.

An automatic step is created after the lead player executes an action that moves the ball to a new position (e.g. pass, forward or dribble).

When using relative positions for players, any positional change made to the position of the players being referenced results, by default, in the re-positioning of the players making the referral as depicted in Fig. 9.

When the user drags a player throughout the field that is used as a reference for the relative positioning of a player, relative position is recalculated based on the player, new position as depicted in Fig. 9(b). However, if the user holds the Shift key while dragging player, relative position to player, initial relative position to player, is preserved and visual feedback is provided to the user who will see also player, accompanying player, movement as depicted in Fig. 9(c). Moreover, relative target actions will always be updated in conformity with the movements of the reference object to whom they are relative to.

When a pass action is defined from a sender to a receiving player who had already been assigned a run action to a given location, the execution of a forward pass is automatically assumed (see Fig. 6) to the receiving player’s predicted final position after the run instead of a direct pass to his initial position at the beginning of the step.

5.1. Defining set-plays using recorded matches

The log viewer embedded in SPLanner can be used to visualize a previously recorded game log of a relevant opponent and simultaneously define a set-play as depicted in Fig. 10.

By using the embedded log viewer the user can observe weaknesses in the opponents positioning scheme in different game situations (e.g. a corner-kick in this example) and try to exploit them using a tailored set-play. Also, using this game log record the time to iteratively tune and test a given set-play against a given opponent for concrete situations is greatly reduced.

5.2. Easing the creation of set-plays

The creation of a set-play in SPLanner has three main phases that consist on:

1. Choosing the type of set-play and the initial situation and region where it will take place;
2. Drag the players into the field and define the actions that allow them to cooperate as intended;
3. Export the set-play for testing purposes or final use.
(a) Choose the type, situation and start position

(b) #4 is automatically placed in the corner

(c) #8 is dragged from the bench to its initial absolute position

(d) #9 is dragged from the bench to its initial absolute position

(e) #9 runs to the entry of the penalty area

(f) #4 passes to #8

(g) #8 dribbles towards the end line

(h) #9 waits for player #8 to complete his dribble

(i) #9 runs to the entry of the goalie area

(j) #8 passes forward to #9

(k) #9 shoots at goal

(l) #9 scores a goal

Fig. 8. Step-by-step example of the definition of a corner-kick set-play with four players.
Fig. 9. Changing the relativity of players positions.

(a) Initial scene in which #3 relative position to #1 is defined

(b) Final scene after moving #1 without holding the Shift key (relative position changes)

(c) Final scene after moving #1 while holding the Shift key (relative position is preserved)

Fig. 10. Defining and tuning of a corner-kick set-play against an opponent (blue team).
Steps and transitions are also automatically created when the leader of a step (assumed to own the ball) performs an action (e.g. pass, dribble) that moves the ball to a new location.

5.3. Enforcing logical constraints in the definition of set-plays

The grammar of the set-play framework is too extensive and allows its users to define a large variety of situations by combining its different elements. However, like many other grammars some of the combinations of its concepts, although syntactically correct can be semantically wrong (e.g. a directive to kick the ball might be specified to a player who does not own it). To circumvent these issues SPlanner limits some of the users decisions to prevent illogical options to be taken. The main measures that were created to achieve the previous goals are:

• A maximum of eleven players can participate in the set-play;
• All players positions must be within the game field bounds, except in particular game situations such as kick-ins and corner-kicks;
• Player actions are always constrained in order for the designed situations to be coherent and logical. For instance, a player that does not own the ball is unable to perform a pass;
• Players that are the intended receivers of a pass should perform a receive ball or intercept action. However, in SPlanner this option is unavailable and it is implicitly added when a pass or forward action are designed;
• Only one transition can be added between each pair of steps in order to preserve the integrity of players actions during the execution flow of the set-play;
• The choice of conditions is always constrained based on the current context and in particular player’s actions. For instance, if a player is going to shoot at goal, a condition to check whether that shoot can be made might be defined. However, this condition will be unavailable for a player executing a pass.

5.4. Simplifying the definition of steps in a set-play

The set-play framework allows a player to perform several actions during a transition to a new step. However, in most cases this is not necessary as simpler steps could alternatively be designed and preserve a similar semantic. For instance in a given step it is possible in the grammar to instruct a player to dribble the ball to a given location and after the dribble pass it to a teammate. In our approach, these actions are performed in two sequential steps, first the dribble and only afterwards the pass.

5.5. Estimating automatically the abort times for steps in a set-play

The set-play framework allows the specification of abort times (see Section 3) for each different step. Since a step might contain more than one transition and each transition might contain several directives that specify for different players the actions that they should perform, the abort time is defined as the maximum time a player will require to complete all of its actions in one of the transitions of that step as described in Algorithm 5.5.1.

Algorithm 5.5.1. estimateStepAbortTime(step)

```java
Require: step ≠ 0
1: abortTime ← 0
2: for each participant in getParticipants(step) do
3:    for each transition in getTransitions(step) do
4:       transitionAbortTime ← 0
5:       directives ← getDoDirectives(participant, transition)
6:       for each directive in directives do
7:          for each action in getActions(directive) do
8:             transitionAbortTime ← transitionAbortTime + estimateDuration(action)
9:          end for
10:     end for
11:    if abortTime < transitionAbortTime then
12:       abortTime ← transitionAbortTime
13:    end if
14: end for
15: return abortTime
```

This feature is very useful to the user as it relieves him from the burden of trying to figure out which would be the adequate abort times, minimizing errors in calculations, for each step of a set-play. The estimation of these timings is done resorting to the physics
model incorporated in the Soccer Server simulator applied to players and the ball. Currently, for simplicity, it is assumed that players are homogeneous and thus are characterized using the same model, although in reality they can have different characteristics which the Soccer Server is already capable of assigning for a match.

These automatically calculated values are not strict and the user can refine them if necessary.

6. Tool validation methodology

In order to validate the developed tool four experiments were performed.

The first experiment consisted on testing the correctness and robustness of the SPlanner import process and consequently the visual representation of previously defined set-plays. The set-plays used in this test have been previously exported from SPlanner or manually defined by FC Portugal team members using a text editor.

The second experiment consisted on measuring the effectiveness of two of the previously created set-plays (a goalie catch and right-corner kick) with SPlanner by playing several matches using the FC Portugal team with and without an opponent. In the context of this experiment, effectiveness refers to the execution of a set-play as expected, based on its definition as perceived from the visual representation provided by SPlanner. It must be pointed out that in order to be effective, it is not required that a set-play ends with the scoring of a goal because its objective could be to simply carry the ball to a given location (e.g. a goalie catch that tries to get the ball as close to the midfield as possible using a wing). It is also possible that the execution of a set-play fails due to one of its abort conditions.

The third experiment consisted on measuring the performance of the execution of tasks related with the creation of set-plays. A group of forty two users (twenty nine with basic computer skills, three members of the FC Portugal team and ten experts in the sports domain) were invited to perform a script with seven typical tasks performed when creating a set-play (manually using a text editor and SPlanner) with different levels of complexity. Only the FC Portugal team members have had a previous contact with SPlanner and thus a baseline needed to be established for all users in order to allow a fair comparison of the results. To achieve this a brief session was privately held with each user in order to explain them the concepts associated with the definition of set-plays in the context of the set-play framework and also the main organization and most common usage patterns of the SPlanner interface. For each task two measurements were made: time elapsed between the end of the task reading and comprehension until it was correctly completed (execution time) and the number of executed actions that were unnecessarily executed, that is which were not aligned with the goal of the task (number of errors).

The tasks contained in the script that was presented to users were:

1. Define an action for a player;
2. Redefine the name of a player;
3. Create a transition to a new step;
4. Build a complete set-play with three participants and three steps which starts from a corner-kick situation in a specific side with absolute positions;
5. Build a complete set-play with two participants and two steps which starts from a kick-in situation in a specific location using relative and undefined positions;
6. Import an existing set-play in order to make some modifications and afterwards export it;
7. Import a previously recorded game log and manage its visualization while defining a set-play.

Tasks one to three were executed in the context of an existing set-play definition previously provided to the users. Tasks four and five were started with the SPlanner tool open in its main screen without any initial definition of a set-play. For tasks six and seven no comparison can be made with the manual process as no similar alternative is currently available.

After the completion of all the tasks, a System Usability Scale (SUS) questionnaire [27] was given to users to assess their satisfaction with the system in a simplistic and subjective manner. This questionnaire was complemented with a biographical questionnaire to assess the users' familiarity with the English language, level of expertise using computers, experience with soccer video games, knowledge of the soccer domain and experience with soccer tactical panels. The SUS questionnaire was comprised of the following ten statements in which positive and negative are intercalated every two statements:

1. I would enjoy using this tool in regularly;
2. I found the tool unnecessarily complex;
3. I found the tool easy to use;
4. I would require technical support to use the tool;
5. The tool functionality is well integrated;
6. The tool functionality has many inconsistencies;
7. Most people would learn to use this tool quickly;
8. The tool was very hard to use;
9. I felt confident using the tool;
10. I needed to learn several things before being able to use the tool.
In order to evaluate the previous statements a Likert scale was used. The Likert scale [28] is a well-known method for scaling responses to survey researches. Respondents are asked to specify their level of agreement on a symmetric scale of a given range for a series of statements of equal relevance. The range of the scale captures the intensity of the respondents’ feelings towards the presented statements. In the context of this work, a scale with five levels of agreement (strongly disagree, disagree, neither agree nor disagree, agree, and strongly agree) with values of one to five respectively was used.

To ease the interpretation of the SUS questionnaire, users’ satisfaction was recorded in a scale from 0 to 100. This scale was created by applying a transformation to the Likert values associated with the answers given by the users. The satisfaction score for a user is calculated using Eq. (1).

$$\sum_{i=1}^{10} 2.5 \times \text{score}(i)$$

where \(\text{score}(i) = \begin{cases} \text{LikertValue}(i) - 1 & \text{if } i \text{ is even} \\ 5 - \text{LikertValue}(i) & \text{if } i \text{ is odd} \end{cases} \)

In order to try and generalize the results to a population the Mann–Whitney test was used with a confidence interval of 95%, being the recommended level of statistical significance of \(p < 0.05\) [29].

7. Results and discussion

7.1. Robustness of the set-plays import process

In the first experiment, the import process of all the set-plays that had been previously developed in SPPlanner was always executed without any errors. When trying to import some of the set-plays defined manually by FC Portugal team members some exceptions occurred. The cause of these exceptions was identified as the definition of multiple actions for a player in a step transition in these set-plays. As previously mentioned in Section 5, some measures were implemented in SPPlanner to simplify the process of defining a set-play by deliberately ignoring some of the possibilities of set-play grammar, such as the previously identified cause for the exceptions. Nonetheless, the tool displayed warning messages referring to the previously identified cause and suggestions to correct the situation, but it was unable to complete the import process. This is a topic for future work, as the tool should be able to anticipate these situations and adjust the imported definitions of such set-plays in conformity with the implemented simplifications. On the other hand, the remaining set-plays that had been manually defined were successfully imported and their visual representation was accurately rendered. The rendering of these set-plays contributed to a more intuitive comprehension of their execution flows, which was much more difficult to grasp from simply reading their written definitions.

7.2. Effectiveness of the developed set-plays

As expected, in the second experiment the goalie catch and right corner-kick set-plays achieved different results depending on whether opponents were used. When no opponents were up against the FC Portugal team, the set-plays went exactly as planned and an entire execution path was followed. However, when playing against an opponent two situations arose that did not allow for a set-play execution path to be completed but led to its termination as expected based on one of the abort conditions specified for the set-plays:

- The opponent team intercepted the ball and the set-play was aborted, as defined in its abort conditions;
- The transition into a new step took longer than it was initially expected in the set-play definition and it was immediately aborted.

In particular, the goalie-catch never scored any goal as its objective was simply to get the ball to the teammate nearest to the midfield. However, the same did not occur with the corner-kick set-play which was tuned to work against a specific team (Bahia2D) using the game log viewing functionality and scored a goal in 75% of its twenty executions.

7.3. SPPlanner usability

In the third experiment, the manual writing of set-plays using a text-editor was initially intended to be done by all users. However, after verifying that the execution times from the most acquainted users, the FC Portugal team members were very high, it was decided that other users would require at least 2 h to perform them which would make it impossible for them to voluntarily perform these tests.

Some of the obtained results were grouped in different user segments (e.g. FC Portugal members, sports users) to facilitate their contextual analysis. The average execution times and the number errors committed during the execution of the proposed tasks by
users are presented in Table 1. The users satisfaction measured after completing the tasks by the SUS questionnaire is described in Table 2.

Non FC Portugal users required on average an additional 4 min and 20 s to complete the whole script of tasks with SPlanner in comparison with FC Portugal users. This was expected due to the higher familiarity of the FC Portugal users with the set-play framework and the development of SPlanner.

With regard to the execution of the tasks in SPlanner rather than manually using a text editor (tasks 1 to 5) by FC Portugal users, a reduction of 90% in the time required to complete them was achieved. The most notorious reduction of times (more than 92% of the time required for its manual counterpart) was achieved in the execution of the simple tasks one and three and the complex task four. The lower times for completing task five when compared to task four when using SPlanner are justified by the fact that after completing task four, users had learned to avoid most of the errors committed in that task.

Furthermore, non FC Portugal users executing the tasks in SPlanner have outperformed FC Portugal users that executed the tasks manually having required minus 84% (approximately 27 min) of the time. This result shows that SPlanner enables all users, even non-experts, to complete the tasks faster than any FC Portugal team member would using a manual editing process.

With regard to the knowledge of the soccer domain, it was inferred that users with good or better knowledge of the soccer domain took less time than users with a worse knowledge ($p = 0.025$). These users spent on average approximately minus 3 min and 20 s than the remaining, a result that was expected since users with a better understanding of the soccer concepts could more easily relate with the tasks and with SPlanner.

With regard to the familiarity with soccer tactical panels, it was observed that users with fewer experience required on average approximately 7 min and 30 s to complete all the tasks, which stands for approximately 25 s more than the time required by more experienced users. This time difference is negligible and tells us that the metaphors used in the Splanner GUI were perceived by users in a similar manner.

Based on the experience of users with soccer video-games, it was possible to infer that users with a good experience took less time (approximately minus 2:30 min) to complete the tasks than users with a worse experience ($p = 0.025$). We believe that this result is justified by the similarity that the Splanner GUI has with the strategy panels that are embedded in soccer video-games.

With regard to the users computer skills, it was possible to infer that users with more than basic computer skills had a better performance than users with worse skills, having required less time to complete the tasks ($p = 0.004$) and also committed fewer errors ($p = 0.006$).

Based on the errors and remarks collected from the users during the execution of the tasks it was inferred that some parts of the interface still need to be perfected. For instance, the higher number of errors committed in task five (see Table 1) was caused by the sub-task of performing a forward pass to a player which was not intuitive to most of the users.

The average user satisfaction results based on the scoring scale defined in Section 6 was of 76.79 ± 9.5 (out of 100), having the most and the least satisfied users given a score of 93 and 58 respectively. Based on the results presented in Table 2 it can be concluded that the majority of users enjoyed using SPlanner because most of the scores per question are near the optimal. However, question ten (see page 29) stands out from the rest and despite its score is not negative it tells us that users had

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average execution times (in seconds) and number of errors committed when performing set-play related tasks by three types of users.</strong></td>
</tr>
<tr>
<td>Task</td>
</tr>
<tr>
<td>Non-sports related users</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Errors</td>
</tr>
<tr>
<td>Sports related users</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Errors</td>
</tr>
<tr>
<td>FC Portugal team members</td>
</tr>
<tr>
<td>Time</td>
</tr>
<tr>
<td>Errors</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average satisfaction and standard deviation of the satisfaction scores (0 to 10) given by users on the SUS questionnaire. The optimum score for each question based on the proposed satisfaction scale would be a 10.</strong></td>
</tr>
<tr>
<td>Question</td>
</tr>
<tr>
<td>Satisfaction</td>
</tr>
</tbody>
</table>
to make a big effort to learn how to use the system and thus improvements must be implemented to ease the learning process. It was also possible to infer that users with better computer skills were more satisfied than users with worse skills ($p = 0.023$). This result was expected since skilled users are more at ease to learn how to use new software. Also, with regard to the experience with soccer video-games, it could be inferred that users with more experience enjoyed using SPlanner more than others with less experience ($p = 0.047$). This conclusion is supported by the fact that the SPlanner GUI was developed based on the tactical panels available on soccer video-games, which gives the tool a more appealing look and invites game users to be more at ease to explore it.

8. Conclusions

The interaction with SPlanner was not equal for all test users but, as demonstrated by the results, users were greatly satisfied with its use having ranked it with an average score of 77 (out of 100). The use of this tool also allowed typical users (FC Portugal team members) to significantly reduce the time required (up to 90% on average) to perform set-play related tasks. Furthermore, users with little knowledge of the domain were able to use the tool with relative ease thus widening the range of target users that can assist the FC Portugal team in future competitions.

The reuse of previously defined set-plays not defined with SPlanner was also possible in most cases and it provided an accurate visual representation of its possible execution flows, which was much more difficult to grasp from the examination of their written definition.

The tool is currently optimized for the definition of offensive set-plays, rather than defensive ones, because these are most likely to be advantageous and successful as the ball is in control of the team that is deploying them. Some compromises were made in order to make the tool more usable which ended up not using the set-play grammar to its full extent and thus it does not allow the definition of some situations. Examples of such compromises include the impossibility to add several actions for a player to execute in a given step (e.g. pass and run), which was assumed to make the graphical interface unnecessarily confusing at the moment.

Special precautions were taken to improve the modularity of SPlanner in order to ease its integration with other strategic modules in the future.

Further developments are possible and were identified for this tool.

One of these developments consists on making the user interface more clear and intuitive. For instance, the visual distinction between start step, intermediate steps and end steps should be made more clear using different color schemes, instead of a simple contour.

Extra functionality regarding the definition of set-plays should be added to SPlanner in order to make it more complete. Some examples of such functionality include but are not limited to:

- Repositioning players in steps other than the first while making the necessary adjustments to their actions. For instance, if a player not participating in the set-play is added to a step other than the first, the tool should assume the specified position as the initial position for that player in the first step. Also, if the player already exists in the previous step and his previous action consisted of a run, by moving the player in the current step the target run position should be changed accordingly;
- Implementing support for the drawing of regions based on the context of the soccer domain (e.g. 2D simulated soccer) to be used in the specification of conditions and target locations for players actions;
- Enable players participating in a step to execute more than one action (e.g. pass and run), using a sequential numbering scheme starting at one to establish the order in which actions should be executed;
- Run an animation for a specific flow of a set-play as an overlay and in parallel with a previously recorded game log to give the user a more accurate feedback on what would happen if that set-play were to be executed in a particular situation that occurred in that game log;
- Add the support for the definition of defensive set-plays, adding for instance the concepts of player defensive barrier and marking. However, the actions to support the creation of these types of set-plays must still be implemented;
- Improve the use of opponent players in order to allow the definition of constraints to the designed set-play in order to make it more robust;
- Automatically infer the actions that should be executed when defining a transition between two existing steps. These type of automatic transitions are very useful because they provide an alternative path for the execution flow of a set-play and contribute to improve its robustness;
- Support undo and redo functionality in order to quickly recover from errors committed when defining a set-play.

In order to make this tool usable for other teams and leagues, an independent plug-in should be developed with its context specificities (e.g. maximum speeds, field size, number of player).

Last but not least, other strategy modules such as the definition of team formation and tactics should be integrated with SPlanner (e.g. Matchflow) to further improve its usefulness. An added value will be created as formations and tactics might be used to determine which set-plays are more adequate to use against a given opponent.
Appendix A. Example set-play definition of a corner-kick

```prolog
(setplay :name ck-right-3p :id 1 :invertable true :version splanner_1.0
   :comment (Corner-kick from the right side with 3 participants)
   :players (list (playerRole :roleName Player4)
          (playerRole :roleName Player8)
          (playerRole :roleName Player9))
   :abortCond (or (bowner :players (player :team opp :number 0))
            (not (playm play_on))
            (not (playm ck_our))))
 :steps (seq
   (step :id 0 :waitTime 0 :abortTime 28
      :participants (list (at Player4 (pt :x 52.5 :y 34))
            (at Player8 (pt :x 38.5 :y 22.5))
            (at Player9 (pt :x 33.5 :y -5)))
   :condition (and (playm ck_our)
      (bpos :region (regNamed :name right))))
   :leadPlayer Player4
   :transitions
   (nextStep :id 1
     :condition (canPassPl :from Player4 :to Player8))
   :directives (list
          (do :players Player8 :actions (intercept)))
          (do :players Player9
               :actions (pos :region (pt :x 33.5 :y 0.5)))
          (do :players Player4
               :actions (bto :players Player8))))
   (step :id 1 :waitTime 0 :abortTime 38
      :participants (list (at Player8 (pt :x 38.5 :y 22.5))
            (at Player9 (pt :x 33.5 :y 0.5)))
   :condition (and (playm play_on) (bowner :players Player8))
   :leadPlayer Player8
   :transitions
   (nextStep :id 2
     :condition (canPassReg :from Player8
          :to (pt :x 44 :y 5.5))
     :directives (list
          (do :players Player8
               :actions (dribble :region (pt :x 50 :y 22.5)))
          (do :players Player9 :actions (stop))))
   (step :id 2 :waitTime 0 :abortTime 32
      :participants (list (at Player8 (pt :x 50 :y 22.5))
            (at Player9 (pt :x 33.5 :y 0.5)))
   :condition (and (playm play_on) (bowner :players Player8))
   :leadPlayer Player8
   :transitions
   (nextStep :id 3
     :condition (canPassReg :from Player8
          :to (pt :x 44 :y 5.5))
     :directives (list
          (do :players Player8
               :actions (bto :region (pt :x 44 :y 5.5)))
          (do :players Player9 :actions (intercept))))
   (step :id 3 :waitTime 0 :abortTime 24
      :participants (at Player9 (pt :x 44 :y 5.5))
   :condition (and (playm play_on) (bowner :players Player9))
   :leadPlayer (playerRole :roleName Player9)
   :transitions
   (nextStep :id 4 :condition (canShoot :players Player9)
     :directives (do :players Player9 :actions (shoot))))
   (step :id 4 :waitTime 0 :abortTime 0
      :participants (at Player4 (pt :x 52.5 :y 34))
   :condition (playm play_on) :leadPlayer Player4
   :transitions (finish))))
```

References

Fernando Almeida is a Lecturer at the Polytechnic Institute of Viseu, Portugal. He is currently taking his PhD which focuses on Automatic Plan Extraction, Recognition and Optimization from collective sports games, with a particular emphasis in soccer. His other interests include (but are not limited to) Artificial Intelligence, Autonomous Agents, Multi-Agent Systems (MAS), COORDINATION in MAS, Automated Reasoning and Inference, Game Analysis and Robotic Soccer.

João Cravo holds an MSc in Informatics Engineering and Computation from the Faculty of Engineering of the University of Porto. He is currently working at Blip.pt, a software company located in Porto, Portugal, and as a Build Master. His other interests include (but are not limited to) Artificial Intelligence, Software Engineering, Graphical User Interface Design and Robotic Soccer.
Pedro Henrique Abreu is an Assistant Professor at Coimbra University, Portugal. He obtained his PhD in soccer teams’ Modeling from the University of Porto (2011). His interests include (but are not limited to) Artificial Intelligence, automatically analysis in Game Analysis, Tactical Modeling, and Data Mining Techniques Applied to sport collective games.

Luis Paulo Reis is an Associate Professor at the University of Minho, member of the Directive Board of LIACC Laboratory and coordinator of the Human–Machine Intelligent Cooperation Research Group in Portugal. He was principal investigator of more than 10 research projects in the areas of Artificial Intelligence and Robotics including FC Portugal, three times World Champion and eight times European Champion at RoboCup. He also won more than 30 other scientific awards. He supervised 13 PhD theses and 80 MSc theses to completion and is the author of more than 250 publications in international conferences and journals. He is the president of the Portuguese Robotics Society.

Nuno Lau is an Assistant Professor at Aveiro University, Portugal. He got his Electrical Engineering Degree from Oporto University in 1993, a DEA degree from Claude Bernard Lyon 1 University in 1994 and the PhD from Aveiro University in 2003. His research interests include Intelligent Robotics, Artificial Intelligence, Multi-Agent Systems and Simulation. He has lectured courses at PhD and MSc levels on Distributed Artificial Intelligence, Intelligent Robotics, Computer Architecture, Programming, etc. Nuno Lau has participated, often with the coordination role, in several research projects that have been awarded international prizes. Nuno Lau is the author of more than one hundred publications in international conferences and journals.

Luís Mota is an Auxiliary Professor at ISCTE — IUL, a university in Lisbon. He got his BSc and MSc from Instituto Superior Técnico (IST, Lisbon) and his PhD from the University of Porto. He has been researching mainly in the MAS and Robotic Soccer areas.