Abstract—We propose probabilistic/statistical model checking as a tool for game design. Models of games under design can be used for the early evaluation of properties related with game duration, existence of different successful strategies, and balancing of the game core mechanics. To show the usefulness of model checking in game design we consider and develop three case studies in which model checking is used to answer specific game design questions.

I. INTRODUCTION

Game design “is the act of deciding what a game should be” [1]. These few words hide the real number of decisions that have to be taken in order to define all the details of a non trivial game. In fact, on the one hand designing a game does not require any special equipment: it is only decision making. On the other hand, in order to design a game that allows players to have a satisfactory experience it is important that the core mechanics of the game are well thought, and that they are well balanced [2].

Game design is one of the phases of game development. The game to be developed can be of many different kinds: a video game, a board game, a card game, an educational game, etc. However, the principles for the design of a good game are often the same independently of the game technology used.

The definition of the core mechanics of a game is the most important part of the game design. The core mechanics are the essential procedures and rules of the game: they describe the goal of the game, how the players can and cannot try to achieve it, and what happens when they try. In order for a game to be satisfactory for the players (or replayable) the game core mechanics have to satisfy a number of properties (some of which are discussed in Section III) that essentially deal with equilibria between the different game aspects. For example: the level of randomization, the existence of alternative winning strategies, and the degree of player cooperation/competition are aspects for which their role in the game should be tuned.

The evaluation of the properties satisfied by a game under development is often not easy. Games can be tested from the early stages of development by means of prototypes, but testing can require time, the involvement of many people, and sometimes give results that are not completely objective. A contribution to the evaluation of a game under development could be given by modelling techniques. Constructing a model of the core mechanics of a game can allow some properties of the gameplay to be precisely evaluated. Aspects such as the game duration, the probability of victory of a player, the probability of victory of different strategies, and the role of each component of the game in the determination of the winner, can all be determined on a model of the game.

In this paper we propose probabilistic/statistical model checking as a game model analysis tool for game design. In particular, we consider three case studies of design of board games and we show how model checking can help in taking decisions on some specific design aspects. We focus on board games since in many aspects they are very similar to computer games. Moreover, the use of board games allow the case studies to be rather simple to explain.

The first case study deals with aspects of game randomization. The second one deals with the existence of different winning strategies in a (competitive) strategy game. Finally, the third one deals with aspects of collaboration in a board game. In each of the three case studies a specific design question is answered by means of model checking. As probabilistic/statistical model checker we adopt the well-established PRISM tool [3]. The source code of the models we have developed are available at http://www.di.unipi.it/msvbio/

II. PROBABILISTIC AND STATISTICAL MODEL CHECKING

Model checking [4] is a technique for the verification of properties (expressed as logical formulae) holding in a system, based on the thorough exploration of its state space. Therefore their applicability is limited to finite-state systems; unfortunately in practice for many interesting systems the state space (albeit finite) is just too big to be examined, therefore various techniques have been developed to tackle this problem [4].

From a theoretical point of view, in the setting of model checking, the dynamic behaviour of a system is typically described by means of a Kripke structure, capturing both the states of the system and how it evolves from one state to another, and which “facts” hold in each state. Formally, properties are expressed in a temporal logic, such as the Computational Tree Logic (CTL), constructed from a basic set of atomic proposition \(AP\) which are composed through the operators provided by the logic. In the qualitative setting, a the behaviour of a system is described by a Kripke structure \(M\) over \(AP\), which is a tuple \(M = (S,S_0,R,L)\) where: (i) \(S\) is the finite set of states; (ii) \(S_0 \subseteq S\) is the set of initial states; (iii) \(R \subseteq S \times S\) is the transition relation; (iv) \(L : S \rightarrow 2^{AP}\) is the labelling function associating, to each state, the set of atomic propositions which are true in that
state. The transition relation can describe non-deterministic behaviours, and it is assumed to be total (\(\forall s.\exists s'.(s, s') \in R\)). Executions of the system are represented by the infinite paths 
\[ \pi = s_0, s_1, s_2, \ldots \]
where \(s_0 \in S_0\) and \(\forall i.(s_i, s_{i+1}) \in R\). In this paper we are mostly interested in analysing probabilistic systems described by means of Discrete Time Markov Chains, for which a probabilistic distribution is associated with the transitions exiting from a state.

Temporal logics are used to describe sequences of transitions between states in the Kripke structure. In the non-probabilistic setting, we consider the Computation Tree Logic (CTL), which is composed of path quantifiers (A, E) and temporal operators (X, F, G, U). Path quantifiers are used to deal with branching in the Kripke structure, i.e. the different transitions exiting from a state: A \(\varphi\) means that \(\varphi\) holds for all paths starting from the current state, while E \(\varphi\) means that \(\varphi\) holds for some paths starting from the current state. Temporal operators describe properties of paths: X \(\varphi\) (next time), \(\varphi\) holds in the “next” state of the path; F \(\varphi\) (eventually), \(\varphi\) holds in some state of the path; G \(\varphi\) (globally), \(\varphi\) holds in all states of the path; \(\varphi_1 U \varphi_2\) (until), \(\varphi_2\) holds in some states of the path and, in all the states which precede it, \(\varphi_1\) holds.

In this paper, we make use of a probabilistic extension of the CTL, called Probabilistic CTL (PCTL), which provides operators to query about quantitative properties of probabilistic systems. In particular, we are interested in the P operator, used to query about the probability of different executions. Among the different forms, we consider queries of the form \(P_{s_0}[\varphi]\), which gives the probability that the evolution of the DTMC follows a path which satisfies the path property \(\varphi\) containing temporal operators. As an example, consider \(P_{s_0}[F \text{player1\_wins}]\) which queries the probability that the atomic proposition \text{player1\_wins} eventually holds, by considering all possible executions. In the probabilistic setting, it is also possible to associate rewards to states and transitions of a DTMC. For each path, a total reward can also associated with the path itself by summing up all the reward of each state and transition along the path (except the final state).

We use the PRISM probabilistic model checker [3] to model and analyse the gameplay of the different games considered. The PRISM model checker is a mature tool which provides both exact model checking algorithms, based on iterative algorithms for approximating the measure of paths satisfying the formula being analysed, and algorithms for statistical model checking, which are instead simulation-based. In statistical model checking [5], many random simulations are performed, and used to derive the probability measures of different system’s executions, with some non-zero risk of erroneous results. However, simulation-based methods are far less demanding than exact methods, and moreover the probability of erroneous results can always be bound by any small amount.

III. PROPERTIES OF “REPLAYABLE” GAMES

A great game is a game that players play again and again. Hence, in particular in the context of board (and video) games, replayability is the property looked for by game designers. An analysis of the characteristics of a replayable game has been recently published on “The Ministry of Board Games”, one of the major blogs specialized on board games (see http://www.mofbg.co.uk/2014/09/lessons-on-re-playability.html).

Some of the main properties to be satisfied by a board game in order to be replayable are related with the following aspects.

Game duration: Game duration is an essential aspect to be considered during game design. Such a duration has to be estimated during game design and used as a key parameter for design choices. As side aspects we mention also (i) the time of determination of the winner and (ii) the time of elimination of the first player from the game (if possible). It is important to ensure that the winner or a loser of the game is not determined (even probabilistically) too early in the game play.

Randomized play: Games can include different levels of randomization. On the one hand there are games that are completely determined by random events (e.g. the Game of the Goose). On the other hand, there are games that are completely determined by player choices (e.g. Chess). Most of the games include both aspects. Hence, during game design it is important to evaluate the role of the different random sources (dice, cards, etc.) in the determination of the winner. In particular, it should be checked whether there is a correct equilibrium between random sources and player strategies, and also between the different random sources. For example, it should be checked that the probability to win does not correlate with the probability of being “lucky” with only one of the sources of randomness in the game. In facts, this would mean that the other random sources are not relevant for the determination of the winner.

Different Strategies: Games that are mainly based on player strategies should ensure that there exist different successful strategies. Moreover, it is often desirable to let the player choose among different alternative strategies, which could be changed or adapted during the game play on the basis of random events or on the behaviour of the other players. Games in which there is a single winning strategy become very soon boring (hence, not replayable).

Competition and Collaboration: Traditional games are competitive: one of the player wins and the others lose. In the last decades, instead, collaborative games have started to be proposed. Roughly speaking, a collaborative game is one in which either all of the players win, or all of them lose. Examples of collaborative games are Pandemic and Zombiecide. The dynamics of collaborative games is very different from that of competitive ones. Collaborative games tend to favour socialization and communication between players, and this is why they are sometimes preferred. Of course, often games are not always completely competitive or collaborative. The inclusion of some degree of competition in a collaborative game, and vice versa, is something that could be considered and evaluated during game design. Moreover, a typical problem of collaborative games is related with the early elimination of one player from the game (already discussed above). The game mechanics should be carefully designed in order to reduce as much as possible the probability of such an event.
Most of the properties of replayable games described above deal with quantitative aspects of the game dynamics such as timing, probability of events, and probability of reaching good/bad game configurations. These are aspects that can be dealt with by applying probabilistic/statistical model checking on models capturing the dynamic behaviour of the system (i.e., game). The use of model checking during game design could hence allow the designer to better evaluate the replayability of the game under development by experimenting on the model the effects of changes to the game mechanics.

In the case studies described in the following sections we will demonstrate in three examples of very different kinds of games how model checking can be used to improve replayability properties during game design.

IV. CASE STUDY 1: GAME OF THE GOOSE

Description of the game: The Game of the Goose is a well-known board game consisting of a track of consecutively numbered spaces (usually 63 and arranged in a spiral). Each player has a piece that is initially placed before the first space of the track (corresponding to position number 0). On each turn, a player rolls one (or two) dice and advances his piece along the track by as many spaces as the points given by the dice. The first player who successfully lands exactly on space 63 is the winner. If a player exceeds the number required to exactly reach space 63, he has to step forward into 63 and then move back by the surplus number of points.

Some of the spaces in the board are usually associated with special behaviours. In this case study we consider only one kind of special spaces, namely spaces that cause the player to move back by a fixed number of positions.

Aim of this case study: The Game of the Goose is a board game that is completely driven by randomness. The dice is the only source of randomness, but it is involved in three different mechanics: (i) progress in the track, (ii) landing on special spaces, and (iii) landing (exactly) on space 63. A regards (i), the luckiest player is the one who always rolls 6. As regards (ii), the luckiest player is the one who never rolls a number that makes him reach a special space (that would move him back). Finally, as regards (iii), the luckiest player is the one who, when is one dice roll far from the end of the track, lands on space 63 at the first attempt.

Let us now play the role of the designer of the Game of the Goose. It would be desirable to tune the three mechanics of the game in order to find an equilibrium that makes all of them equally important to determine the winner of the game. In particular, let us focus on mechanics (i) and (ii). It would be important to make sure that making high rolls and avoiding special spaces are equally important in order to win the game.

The role of the special spaces can be modulated by varying their number, position on the track and effect on the player who lands on them. We focus on the first two parameters (number and position) by assuming, as the only effect, the backward movement of the player by six spaces. By means of probabilistic model checking, in this case study we look for the smallest number of special spaces to be included in the track in order affect the determination of the winner.

PRISM model: The model is composed of modules $M_{\text{Player1}}$ and $M_{\text{Player2}}$, for the two players, and module $M_{\text{Turns}}$ which alternatively enables $M_{\text{Player1}}$ and $M_{\text{Player2}}$ to advance, thus implementing change of turn between players. The model is parametric w.r.t. the position of special spaces, identified by the constants $p_1, \ldots, p_{10}$. The effect of a player turn is described in two steps: first (i) a probabilistic transition describing the change of position along the track following the dice being thrown (also including the possible backward steps if it would advance beyond the end of the track), and then (ii) a transition to model the effect of special spaces. For $M_{\text{Player1}}$, the position along the track is described by a variable $x \in [0..63]$. Moreover, the number of times the player falls on special spaces is recorded in variable $cpx \in [0..\text{max penalties}]$, where $\text{max penalties}$ is a constant (set to 10) which ensures that no single player is penalized more than a fixed number of times, hence making the model finite. The value $\text{max penalties} = 10$ has been chosen because, in any of the settings considered, the probability of falling in a special space more than 10 times is negligible. For $M_{\text{Player2}}$, such variables are called $y$ and $cpy$, respectively. Module $M_{\text{Turns}}$ exploits the synchronization feature of PRISM to ensure that the two player modules are enabled alternatively. The following formulae are defined to handle termination: $p_{1\text{\_wins}} \equiv (x = 63); p_{2\text{\_wins}} \equiv (y = 63); \text{terminated} \equiv p_{1\text{\_wins}} \lor p_{2\text{\_wins}}$.

Property verification and game design: We initially consider a version of the game that includes no special spaces, and with player 1 starting first. The following properties have been tested, using probabilistic model checking.

\begin{align*}
A & [\text{F terminated}] = \text{false} \quad (1) \\
P_{a_2}[\text{F terminated}] = 1.0; \quad (2) \\
R[\text{turn\_count}]=[7][\text{F terminated}] = 19.7466 \quad (3) \\
P_{a_2}[\text{F p1\_wins}] = 0.5317; \quad (4) \\
R[\text{loser\_position}]=[7][\text{F terminated}] = 57.999 \quad (5)
\end{align*}

Queries (1) and (2) tell us that although in principle there could be non terminating game plays, the game terminates with probability 1. Query (3) is used to compute the expected length of a game play, which is 19.7466 turns. Queries (4) give us the probabilities of victory for each player. Not surprisingly, the player who starts first has a higher probability of victory than the other player (0.5317 and 0.4683, respectively). Moreover, query (5) tells us that the expected position of the loser (which may be either of the two players) when the game terminates is near space 58. This is important, since it means that by the end of the play, the loser is not far away from finish.

We first investigate the effect of the length of the track on the probability of victory for the starting player. Testing (4) on increasingly longer tracks shows that this probability slowly decreases, therefore cancelling the advantage of the starting player for longer tracks. In particular, the probabilities
Let us now consider the role of special spaces. First of all, we can see in Figures 1a and 1b what happens to the probabilities of reaching each space of the track when a special space is added in positions 20 and 40, respectively. Such figures show that special spaces create alterations in the probabilities of nearby spaces, but they do not affect the probabilities of the final spaces, and in particular of the last one (that is, the probability of victory). A special space can have an influence on the probability of victory only if very close to the end of the track.

The role of special spaces is hence only to add some variability to the game, so that players should not only be lucky with dice rolls for the progress on the track, but should also be lucky in not reaching special spaces.

The question we want to answer now is: How many special spaces should be included so that the winner is most likely the one who fell on the special spaces fewer times? In order to answer this question we computed two conditional probabilities: the probability of victory of the player that fell on the special spaces (strictly) fewer times, and the probability of the winner to be the one that fell on the special spaces (strictly) fewer times. By focusing on the first player, these probabilities can be computed as follows:

\[
P_w \left[ \mathbf{F} \left( p_1_{\text{wins}} \land cp < cp_1 \right) \right] / P_w \left[ \mathbf{F} \left( \text{term}, \land cp < cp_1 \right) \right]
\]

\[
P_w \left[ \mathbf{F} \left( p_1_{\text{wins}} \land cp < cp_1 \right) \right] / P_w \left[ \mathbf{F} \left( p_1_{\text{wins}} \right) \right]
\]

which, for simplicity, we denote as \( P(p_1_{\text{wins}} \mid cp < cp_1) \) and \( P(cp < cp_1 \mid p_1_{\text{wins}}) \), respectively.

Fig. 1. First case study (Game of the Goose). Probabilities of reaching each space, with different positions for the special space. (●: Player 1; ■: Player 2)

are 0.5276, 0.5232, and 0.5188, for track lengths of 126, 252, and 504 spaces, respectively.

Let us now consider the role of special spaces. First of all, we can see in Figures 1a and 1b what happens to the probabilities of reaching each space of the track when a special space is added in positions 20 and 40, respectively. Such figures show that special spaces create alterations in the probabilities of nearby spaces, but they do not affect the probabilities of the final spaces, and in particular of the last one (that is, the probability of victory). A special space can have an influence on the probability of victory only if very close to the end of the track.

The role of special spaces is hence only to add some variability to the game, so that players should not only be lucky with dice rolls for the progress on the track, but should also be lucky in not reaching special spaces.

The question we want to answer now is: How many special spaces should be included so that the winner is most likely the one who fell on the special spaces fewer times? In order to answer this question we computed two conditional probabilities: the probability of victory of the player that fell on the special spaces (strictly) fewer times, and the probability of the winner to be the one that fell on the special spaces (strictly) fewer times. By focusing on the first player, these probabilities can be computed as follows:

\[
P_w \left[ \mathbf{F} \left( p_1_{\text{wins}} \land cp < cp_1 \right) \right] / P_w \left[ \mathbf{F} \left( \text{term}, \land cp < cp_1 \right) \right]
\]

\[
P_w \left[ \mathbf{F} \left( p_1_{\text{wins}} \land cp < cp_1 \right) \right] / P_w \left[ \mathbf{F} \left( p_1_{\text{wins}} \right) \right]
\]

which, for simplicity, we denote as \( P(p_1_{\text{wins}} \mid cp < cp_1) \) and \( P(cp < cp_1 \mid p_1_{\text{wins}}) \), respectively.

Fig. 2. Conditional probabilities expressing the role of special spaces by varying their number on the track.

The results of these computations for different numbers of specials spaces are shown in Figure 2. Note that the distance between consecutive special spaces is at least equal to six spaces, in order to avoid the possibility of falling in the \( n \)-th space after falling on the \((n+1)\)-th. The graph shows that \( P(p_1_{\text{wins}} \mid cp < cp_1) \) is less sensible to the number of special spaces than \( P(cp < cp_1 \mid p_1_{\text{wins}}) \). The intuition for this is that the player that has fallen fewer times on special spaces has an advantage independently of the number of special spaces. On the other hand, by increasing the number of special spaces, since it is more probable to fall on them, it is also more probable that the player who wins is the one who has fallen fewer times on them.

The curve of \( P(cp < cp_1 \mid p_1_{\text{wins}}) \) shows an initial rapid increase, but then reaches a quite stable value for a number of special spaces greater than or equal to 5. The value of the probability that is obtained with 5 special spaces is 0.5954; i.e. in nearly 60% of the cases the winner is also the one who has been luckier with special spaces. Hence, 5 could be a reasonable number of special spaces to be included in the track to make them have a significant role in the gameplay.

V. CASE STUDY 2: A STRATEGY GAME

Description of the game: We consider a simplified strategy game inspired by the board game Risk. Risk is a turn-based board game in which the board contains a map of the Earth divided into forty-two territories and each player owns a number of armies. Territories are initially assigned to players that have to place some armies on each of them. Then, at each turn a player can attack a territory of another player from one of his territories in the neighbourhood. The result of an attack is determined by roll of dice with a mechanism that favours the territory with the higher number of armies and, secondarily, the territory that is under attack.

In our simplified model we consider a board with six territories, arranged as in Figure 3, where nodes are territories and edges represent adjacency. We assume two players. At the beginning, territories S1, S2 and S3 are assigned to player 1, and territories S4, S5 and S6 to player 2. Each territory contains six armies, hence each player owns 18 armies.
At each turn, a player receives $n/2$ extra armies, where $n$ is the number of territories he controls. Such extra armies are added to one of his territories randomly chosen. Subsequently, the player chooses whether to attack (if possible) or to pass the turn, with a probability to attack given by parameters $p_1$, for player 1, and $p_2$ for player 2. In case of attack, the two (adjacent) territories involved are randomly chosen. The attacking player and the player that owns the attacked territory both roll a dice. The attack is successful if the attacker roll is (strictly) higher than the roll of the other player. One army is removed from the territory that loses this competition. In case the attacked territory remains without armies, it is invaded by the attacker that puts one of his armies in it (taken from the attacking territory). The attacking territory cannot remain without armies since attacks are allowed only from territories with at least two armies. Once an attack is completed, the player chooses whether to continue with another attack, again with the probabilities given above. The game is won by the player that conquers all of the territories. Note that the choice of the territory to receive extra armies, and of the one to be attacked are random in order to allow results of the study to depend only on the values of parameters $p_1$ and $p_2$.

**Aim of this case study:** The aim of this case study is to show how model checking can be used during game design to evaluate the properties of different game strategies in the game under development. In particular, by model checking it is possible to evaluate what happens to the gameplay by varying the strategies adopted by the players. We focus on strategies that differ in level of aggressiveness of the player, that roughly correspond to considering different values for parameters $p_1$ and $p_2$. The effects on the gameplay that we measure are related with the duration of the game and the probability of a player to win. Indeed, from the point of view of the game designer it would be desirable for the game to enable different alternative winning strategies, and to avoid strategies that allow one player to make the game to continue forever.

**PRISM model:** Variables $s_1, \ldots, s_6 \in \{0, 1\}$ denote the owner of each territory, with 0 and 1 denoting player 1 and 2, respectively. The following formulae are defined to handle termination: $player_1\_wins \equiv \bigwedge_{i=1}^{6}(s_i = 0)$, $player_2\_wins \equiv \bigwedge_{i=1}^{6}(s_i = 1)$, and $\text{terminated} \equiv player_1\_wins \lor player_2\_wins$.

Module $MPlayer$ implements the player’s behaviour, which steps through up to six different phases each turn. Variable $phase \in [0..5]$, initialized to 0, holds the current phase. In phase 0, the current player receives reinforcements and randomly puts them in one of its territories, with uniform probability. Phase 1 handles the probabilistic alternative between either (i) performing an attack (with probability $p_1$), which causes phase to be switched to 2, or (ii) letting turn change and going back to phase 0. In phase 2, the territory which will be attacked (variable $def \in \{0..6\}$) is randomly selected, with uniform probability among all the territories owned by other player for which there is an adjacent territory from which the current player can launch an attack (i.e. containing at least two armies). In phase 3, the attacking territory is randomly selected, setting the variable $att \in \{0..6\}$. Phase 4 implements the competition between the attacker and the defendant, which causes either of the two players to lose one army from their territories. Finally, in phase 5, if the territory being defended has become empty, it means that the attacker has conquered such a territory, thus one of its armies is moved from territory $att$ to $def$, and phase is set to 1, restarting the cycle.

The number of turns is counted in variable $turn\_counter$. A deadlock condition is added when $turn\_counter = 1000$ in order to avoid an infinite state space.

**Property verification and game design:** Since probabilistic model checking of properties of this case study can take hours, we choose here to follow the statistical approach by performing 1000 simulations for each property to be verified, which gives results in a few seconds. We set a confidence level of 99% for the result of each property.

We start by verifying whether the game always terminates. This is checked by testing property $P_{\forall p_1, p_2}(\text{terminated} \& \text{turn\_counter < 1000})$ for values of $p_1$ and $p_2$ that varies from 0.1 (conservative player) to 0.9 (aggressive player) by steps of 0.2. The results are shown in Table 1, for which the maximum error is about ±0.04. In addition, we computed the average game duration (in terms of turns) in case of termination (detailed results not shown). It emerges that if both players are rather conservative, the game does not terminate with very high probability (1.0 in most cases) or lasts hundreds of

### Table I

<table>
<thead>
<tr>
<th>$p_2/p_1$</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.013</td>
<td>0.837</td>
<td>0.872</td>
</tr>
<tr>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.291</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.017</td>
<td>0.261</td>
<td>0.992</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>0.7</td>
<td>0.831</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>0.9</td>
<td>0.843</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>$p_2/p_1$</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>N/A</td>
<td>N/A</td>
<td>0.385</td>
<td>0.828</td>
<td>0.954</td>
</tr>
<tr>
<td>0.3</td>
<td>N/A</td>
<td>N/A</td>
<td>0.237</td>
<td>0.686</td>
<td>0.820</td>
</tr>
<tr>
<td>0.5</td>
<td>0.706</td>
<td>0.755</td>
<td>0.512</td>
<td>0.610</td>
<td>0.685</td>
</tr>
<tr>
<td>0.7</td>
<td>0.181</td>
<td>0.312</td>
<td>0.435</td>
<td>0.496</td>
<td>0.550</td>
</tr>
<tr>
<td>0.9</td>
<td>0.020</td>
<td>0.166</td>
<td>0.350</td>
<td>0.464</td>
<td>0.504</td>
</tr>
</tbody>
</table>
turns. On the other hand, higher levels of aggressiveness cause shorter (average) game durations, with values that reach ~36.

Now we perform some experiments to try to answer the question: Is there a single successful strategy in this game? Since we focus on strategies that differ on the player aggressiveness, we start by considering different (fixed) values of \( p_1 \) and \( p_2 \). This corresponds to considering different strategies that do not take the configuration of the game in account.

We compare the probability of victory of players with different levels of aggressiveness. In particular, we verify property \( P_{\text{50}}[F_{\text{player1_wins}}] \) by varying \( p_1 \) and \( p_2 \), and we divide such values for the corresponding probabilities of termination computed before. This gives the probability of victory in case of termination, that is, the conditional probability \( P(\text{player1_wins} \mid \text{terminated} \land \text{turn_counter} < 1000) \). The results are shown in Table II.

From these experiments, it emerges that higher levels of aggressiveness correspond to higher probabilities of victory. Let us now consider values for \( p_1 \) and \( p_2 \) that are no longer constant, but depend on the game configuration. In particular, we consider such values to be a function of the number of armies (denoted \( \text{tanks}_1 \)) owned by each player. For instance, we choose to define \( p_1 = 0.9 \), if \( \text{tanks}_1 > k \), and 0.1 otherwise, with \( k \) varying. If we consider the situation in which the first player uses such a “dynamic” strategy and the second player uses the constant value for \( p_2 = 0.9 \) we obtain, for any value of \( k \), that the second player always has a higher probability to win with respect to the first player. (Detailed model checking results are omitted due to space limitations.)

Let us now consider a strategy that changes on the basis of the duration of the game. In particular, let us assume that a player would be conservative for some turns (in order to accumulate armies) and then become aggressive. In this case, for the first player, the definition of the probability of attack becomes \( p_1 = 0.9 \), if \( \text{turns} > k \), and 0.1 otherwise.

If, again, we make such a player play against an aggressive player \( (p_2 = 0.9) \) we obtain the probability of victory depicted in Figure 4; throughout all the runs, the error is about ±0.04. It follows that waiting a few turns (up to ~37) makes victory more probable. Now, since this seems to be a promising strategy, it could be expected that both players adopt it. In Table III, we compare the probability of victory of the first player when both players wait a variable number of turns before becoming aggressive (the error is ±0.04). From these results it emerges a quite variegate situation: the first player has the a higher probability of victory if it either waits very few turns, while the other player waits much longer, or if it waits at least 15 turns while the other waits much fewer turns.

As a consequence, the question on when to become aggressive seems not to have a unique optimal answer. This conclusion supports the hypothesis that there is no unique winning strategy, and players have to dynamically take into consideration the game evolution and configuration.

### VI. Case Study 3: A Collaborative Game

**Description of the game:** We consider a prototype collaborative game in a dungeon crawl scenario. The game is for four players and the board depicts ten rooms to be passed through sequentially by the players. Each of the nine first rooms contains some monsters, in a number that varies (randomly) between 4 and 7. The players have to fight and kill all of the monsters of a room in order to pass (all together) to the next room. Each player starts with a predefined number of health points, and loses one health point every time he loses a combat with a monster. A player that loses all of his health points is excluded from the game.

In the first nine rooms, at each turn one player attacks a monster by rolling two dice. Another dice (rolled by another player) represents the monster’s defence. If the monster’s dice is (strictly) higher than both player’s dice, the player loses one health point, otherwise the monster is killed.

The final room contains a special final monster. It has to be fought by all of the players together that reach such a room. In this case, the monster rolls four dice while each player rolls two dice. The lower dice of each player is discarded and the sum of the others is compared with the sum of the dice of the monster. The players win the game if the sum of their dice is greater than or equal to the sum of the dice of the monster.

**Aim of this case study:** This game is collaborative, in the sense that the players either win all together, or lose all together. Moreover, the players have a higher chance to kill the final monster if they all reach the final room. However, the game is completely determined by random events, that is, by the number of monsters in each room and the dice rolls. Moreover, it may suffer from a typical problem of collaborative games, that is the fact that a player might be excluded quite early from the game, spending the remaining time waiting for the other players to conclude.
The aim of this case study is to improve the game by evaluating additional mechanics that avoid early exclusion of a player and, at the same time, include elements of competition that require the players to choose between collaborating or not with other players during the game.

**PRISM model:** Module MGame implements the core mechanics of the game, by keeping track of the health points of each player in variables \( g_1, g_2, g_3, g_4 \in [0..\text{max\_health}] \), where \( \text{max\_health} \) is the initial number of health points. Variable \( \text{room} \in [1..10] \) holds the current room, while \( \text{turn} \in [1..4] \) denotes the current player. Upon entering a room, the number of monsters is randomly chosen with uniform probability among \( \{4, 5, 6, 7\} \), and stored in the variable \( \text{monsters} \). When \( \text{room} < 10 \), a probabilistic transition is used to model the fight of the current player against a monster, whose probability of victory (~0.745) is precomputed according to mechanics described in the previous section. When all monsters are killed, players move to the next room.

The final fight is modelled in MFinalMonster, containing variables \( \text{monster\_sum} \) and \( \text{players\_sum} \) to hold the sum of the dice rolled by the monster and the players, respectively. This is implemented using eight intermediate steps (variable \( \text{step} \in [-1..8] \)), driven by MFinalMonsterSteps. The following formulae are defined: \( \text{game\_won} \equiv \text{step} = 8 \land \text{players\_sum} \geq \text{monster\_sum}; \text{game\_lost} \equiv (\text{step} = 8 \land \text{players\_sum} < \text{monster\_sum}) \lor (g_1 = 0 \land g_2 = 0 \land g_3 = 0 \land g_4 = 0); \text{terminated} = \text{game\_won} \lor \text{game\_lost} \).

The model is also extended to include a variable \( \text{extra\_health} \in [-1..\text{initial\_extra\_health}] \), which holds the number of extra “lives” which can be given to the players when they finish health points. In this case, the player’s health points are reset to the initial value \( \text{max\_health} \), while \( \text{extra\_health} \) is decreased by 1. A “rescue” mechanics is also implemented, by which a player with no health points can obtain 1 health point, if there is another player with at least 3 health points willing to sacrifice 2 of them to rescue the first player. At each turn, the player probabilistically decides either to rescue another player with probability \( \text{recovery\_probability} \) (if there are some), or to fight a monster.

**Property verification and game design:** By construction, the game always terminates. We evaluated, using probabilistic model checking, the probability of victory with respect to the number of initial health points for each player, with \( \text{max\_health} \) varying from 1 to 10, by checking the property \( \text{P}_{ex\_lives}[\text{F}\text{game\_won}] \). The maximum probability obtained is 0.826, for \( \text{max\_health} = 10 \), and with \( \text{max\_health} = 6 \) we obtained 0.487, which we decided to select as the basis for the subsequent tests. For comparison, for \( \text{max\_health} \) values 5 and 7, the probabilities are 0.280 and 0.653, respectively.

The expected length of the game play, computed with \( \text{R}\{\text{turn\_counter}\} \	ext{F}\text{terminated} \) for \( \text{max\_health} = 6 \) is 65.5 turns, while for greater values it stabilizes at about 66.4. We also computed the expected number of turns in which some player has been excluded, when considering all executions, which reaches the maximum values of 21.1 for \( \text{max\_health} = 3 \), and 20.7 for \( \text{max\_health} = 4 \). It then decreases to 10.0 for \( \text{max\_health} = 6 \), going down to 0.26 for \( \text{max\_health} = 10 \). In other words, for about 1/6 (15.3%) of the duration, at least one player is actually excluded.

To counteract this fact, with the aim of diminishing this value, we considered the two other mechanics introduced before. The results are reported in the following table.

<table>
<thead>
<tr>
<th>Case</th>
<th>P[\text{end 1–3p.}]</th>
<th>P[\text{end 4p.}]</th>
<th>P[\text{victory}]</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>base</td>
<td>0.513</td>
<td>0.391</td>
<td>0.487</td>
<td>15.3%</td>
</tr>
<tr>
<td>ex. lives I</td>
<td>0.087</td>
<td>0.913</td>
<td>0.800</td>
<td>1.25%</td>
</tr>
<tr>
<td>ex. lives II</td>
<td>0.457</td>
<td>0.448</td>
<td>0.509</td>
<td>12.5%</td>
</tr>
<tr>
<td>rescue</td>
<td>0.266</td>
<td>0.581</td>
<td>0.559</td>
<td>8.03%</td>
</tr>
</tbody>
</table>

We first tested the mechanics providing extra “lives” to players by setting \( \text{extra\_health} = 2 \) (case \( \text{ex. lives I} \)). We can see that the probability of victory (computed as \( \text{P}_{ex\_lives}[\text{F}\text{game\_won}] \)) has increased to 0.800, and the probability that the players reach the final room all together is very high (0.913), thus making the game too easy. If we set \( \text{max\_health} = 4 \) (case \( \text{ex. lives II} \)), we see that the probability of victory is similar to the base case, albeit the number of turns played by less than 4 players (column \( \text{Wait} \)) is smaller. On the other hand, if we introduce the “rescue” mechanics to the base case (by setting \( \text{recovery\_probability} = 0.9 \)), we obtain a threefold improvement. In particular, the probability of reaching the final states all together increases to 0.581, thus exceeding half of the game plays, while the probability of victory has increased just a little. Moreover, a big improvement can be found in the percentage of turns played with less than 4 players, which has almost halved with respect to the base case.

**VII. CONCLUSION**

We have shown on three selected case studies that probabilistic and statistical model checking can have a role during game design. We focused on board games, but, since principles of game design are mostly independent of the specific game technology, we believe that model checking could also be used with profit during the design of other kinds of games. In one of the three case studies, the statistical approach has been used. Such an approach made it possible to reason on the possible game evolutions without the performance problems of the probabilistic approach. We have used the PRISM model checker for running the experiments for the case studies. The modelling language of such a tool is very general. As a future work, the development of a modelling framework addressing the specific needs of game design could be considered.

**REFERENCES**