Flexible story generation with Norms and Preferences in computer role playing games

Edward Booth[†], John Thangarajah[‡] and Fabio Zambetta[*]
School of Computer Science & IT, RMIT University
Melbourne (Australia)
Email: [†]s3327858@student.rmit.edu.au, [‡]john.thangarajah@rmit.edu.au, [*]fabio.zambetta@rmit.edu.au

Abstract—**Interactive storytelling is a strength of table-top role playing games as they are facilitated by a game master (GM) who directs the narrative and devises game scenarios. One difficulty with the implementation is the large amount of time, effort and specialist skills that can be required for the creation of such an agent. This paper presents a method for developers to shape the narrative by defining game behaviour in terms of norms and preferences. The system was evaluated with both a case study and a user experiment that showed the users found the system to be both user friendly and suitable for development of games.**

Keywords*—*Game Mastering, BDI reasoning, Role-Playing games.

I. Introduction

Table-top Role Playing Games (RPGs) are a style of game in which play is facilitated by a GM who directs the narrative and creates the scenarios which the players will negotiate [1]. In computer RPGs (CRPGs) the GM feature is not usually present and thus the narrative is directed in a largely predetermined manner by designers specifying the events and their sequence. It is also usual for gameplay environments to be built at development time and remain static for each play through. This results in games for which players have little motivation to replay in terms of narrative whereas, in contrast, an infinite number of possible storylines are possible with a human GM.

In an attempt to address this shortcoming, Luong [2] implemented a system using a table-top style GM agent within the CRPG Neverwinter Nights [3] (NWN). This GM guides the player through a given scenario by selecting a path of actions via which the player can achieve the goal. The GM then instructs the player about what to do and manipulates certain factors in the game world to guide the player through this sequence of actions. For example, the GM agent may select a path involving the player bribing a specific character with a favour to obtain a password. The GM would consequently have that character announce that they need assistance. Due to the number of possible actions and their sequences a large amount of possible storylines for the quest are possible and each play-through is likely to be quite different to the last. Luong's implementation used a BDI agent oriented framework to implement the Neverwinter Nights GM. BDI [4], [5]; Belief-Desire-Intention agents are a popular and mature agent development paradigm based on cognitive concepts such as beliefs, goals, intentions and plans. Agent oriented programming allows for the creation of autonomous agents to perform tasks which are capable of reasoning and reacting to environmental changes.

One shortcoming of Luong's approach is that the GM's selection of storylines is essentially random. Using this approach, game designers would have no control over which of the implemented plot points the GM would present to the player. Another issue with the existing GM is that a game designer would need knowledge of agent-oriented and BDI concepts in order to design how game scenarios will function as the game rules are embedded in the BDI implementation.

We address these issues by implementing an innovative application of the well explored concepts of norms and preferences into the GM agent. We use norms and preferences because we are able to specify them separately from the agent code and they can dictate agent behaviour without the need for re-implementation and testing of agent definitions themselves. We also approach preferential reasoning in a novel way by using probability information to give the agent some information about the lower level implications of selecting higher level plans.

A norm is an application of a rule to one or more agents for which a sanction is incurred by an entity who breaks it [6], [7]. These rules can take the form of obligations or prohibitions. For example Amy is obligated to arrive at her job on time and will receive a warning from her employer if she is late. She is also prohibited from driving faster than the specified speed limit and if she exceeds that limit she may incur a fine. When a norm restricts behaviour, it is necessary for some reasoning to occur against overall goals. For instance in the above example, if Amy's friend has been badly injured and it is necessary to get him to the hospital as quickly as possible, Amy may choose to break the speed limit in order to do so as that goal takes precedence over a potential fine.

In this context, preferences are defined as an agent's predisposition to achieve a goal in a certain manner [8]. These preferences come into play during reasoning when the agent has multiple plans it can select to fulfill an objective. For example if Joe's goal is to get to work on time, he may prefer to catch the train to get there. However if Joe misses the train he can still call a taxi. In this case the preference was not satisfied but Joe was still able to achieve the goal.

II. Background and Related Work

BDI Agents: The BDI approach [5], [9] is to model and implement agents using mental attitudes, such as beliefs, desires, goals, plans and intentions. Practically, BDI agent systems enable programmers to write abstract procedures, called plans, that are combined and used in real-time in a way that is both flexible and robust. Briefly, a BDI system responds to events (i.e., inputs to the system), by selecting a plan from a predefined plan library based on the current beliefs. A plan is a set of steps to respond to a particular event. These steps could include actions that affect the environment, belief updates, or internal events (subgoals) that are in turn handled by different plans. When an agent commits to a particular plan it is added on to the intention stack. The system continues this process of sensing, reasoning and acting.

Goals are often represented as (internal) events in BDI implementations. There is often more than one plan in the plan library that can achieve a goal, thus providing 'flexibility'. If the chosen plan fails then the system will choose an alternative plan, if any, to achieve the goal, thus providing 'robustness' upon failure. The fact that plans that handle goals may in turn post subgoals, naturally leads to a goal plan tree structure. In essence, it is an 'AND/OR' tree where one of the plans is chosen to achieve a goal ('OR'), and all of the subgoals must be achieved for the plan to succeed ('AND'). Padgham and Winikoff [12, p.16] show the power of flexibility of these goal-plan structures by providing calculations that illustrate that a modest system with 72 sub-goals and 147 small plans (4 subgoals each) provides over two million ways of achieving the single top level goal.

There have been a number of BDI agent system design methodologies proposed with supporting tools [4]. For the purpose of illustrating our approach we chose the popular Prometheus [10] methodology and the supporting design tool (PDT) [11]. There are 3 main design stages in Prometheus: System specification where the requirements of the system are captured including the goals of the system; Architectural design where the overview of the internals of the system are specified; and Detailed design where each agent's internals are detailed in terms of the goal-plan trees for the goals they achieve. PDT provides a graphical interface for designing a system and provides code generation, amongst many other features, that generate code stubs in JACK [12].

Preferences: Our system uses preferences to describe the kind of game scenario which will be presented to the player. In the planning domain preferences refer to the desirability of certain outcomes over others. These preferences may relate to desired states, actions or overall plan properties and may be ordered and/or dependant on certain conditions. Preferences are typically specified in terms of certain properties which different states and actions satisfy and can be given numerical values to express measures of desirability.

The preference specification language $\mathcal{PP}$ was developed by Son and Pontelli [13] and implemented in an Action Set Planner [14] which represents the planning problem as sets of actions, fluents (time dependent relations), predicates describing interactions between the world and the initial state. The planner then translates the problem into a logical problem described by a set of domain-dependent and domain-independent rules.

$\mathcal{PP}$ was extended by Bienvenu et al [15] to create their planner **PPLAN** by adding quantifiers, variables, non-fluent relations, a conditional construct, and aggregation operators (AgPF). Quantifiers and variables provide support for numeric preferential specification and reasoning. Whereas in $\mathcal{PP}$, preference formulas were ordinal (ie: one preference could be defined as more important to satisfy than another), here the relative differences between preference formulas can be quantified and thus incorporated into reasoning. Preference formulae are given values from a totally ordered set, $\mathcal{V}$, with a minimum and maximum value

Preferences have been previously implemented in a BDI context with similar principles such as in [8], where preferences could be specified in relation to plan properties. These properties provide more detailed information about what exactly will happen when a plan is used to achieve a goal. In the introductory example, a plan involving catching a train to work may set a variable to 100 eg: *transportMethod.train(100)*. Numerical plan property values are then compared to preferences to sort the plans applicable to the goal in terms of how well they suit the preference set. The paper also deals with the upward propagation of plan properties through the tree so that any reasoning about plan properties takes into account the properties of the lower level plans which may need to be carried out in order to satisfy a plan's sub-goals. This issue is also present in the our work and we propagate plan properties (as discussed in Section III-D) in a similar fashion to the method described in [8].

Norms: A norm is a specified rule which, if violated by an agent it is applicable to, applys a sanction to that agent. Whilst our system implement norms endogenously (the agent will enforce its own norm compliance), numerous infrastructures have been proposed for the specification of norms exogeneously (ie: enforced by entities other than the agents they apply to) via organisations [16] or institutions [17], [18].

The formal framework of ISLANDER [17], [19] defines an electronic institution in terms of roles, scenes and norms, in which different types of agents can interact with each other. Here norms are used as a way of specifying interactions which may not be detrimental to the agent. Whilst these norms can specify undesirable outcomes for the agent, this usage contrasts to other interpretations where norms have a sanction attached to them to specifically encourage adherence to an obligation or prohibition.

Another approach is a middleware $\mathcal{S}-\mathcal{M}$OISE$^+$ [16] which facilitated the interaction of $\mathcal{M}$OISE$^+$ [20] organisations with agents developed in any architecture. It specifies both unbreakable hard constraints and soft constraints which agents may choose to violate, however there is no in-built mechanism for dealing with such violations.

A framework called N-2APL was presented by Alechina et al. [6] and incorporates BDI based specifications such as beliefs, goals and plans, with normative concepts such as obligations, prohibitions and sanctions. It allows agents to deliberate on whether to adhere to or to violate norms, including support for temporal factors such as deadlines and durations. However, this reasoning is based on priorities assigned to the agent's goals and to the norms which the agent is subject to as opposed to such factors as the undesirability of the sanction itself and outcomes which would arise from said sanction. A more complete architecture has since been proposed which integrates N-2APL with a norm specification language 2OPL [7] via a JavaSpaces tuple space. This implementation allows norms to be activated (detached) and to expire at run-time and monitors and enforces these norms, applying sanctions in cases of violation.

The prototype CRPG - Neverwinter Nights: Neverwinter Nights is a third-person role-playing computer game developed by BioWare and published by Atari in 2002. The game is set in a fantasy world, with the game rules based on the Dungeons & Dragons 3.0 rule system. NWN includes a game engine, a game campaign (the actual game itself) that can be played as single player or in multiplayer mode, and the Aurora toolset that has made the game extremely popular. It has also been employed in research work [21], [22] since it is still probably the best (almost) freely-available toolset for computer role-playing games to date.

III. THE GM SYSTEM WITH NORMS & PREFERENCES

Our architecture is comprised of a number of different aspects. First there are the norm and preference specification languages which we have developed with the goal of making them human readable and easy to learn. Then there is the game master agent itself, into which the specification languages were integrated, along with additional information and reasoning methods with which the GM can act upon them. Figure 1 illustrates the overall communications between architectural components.

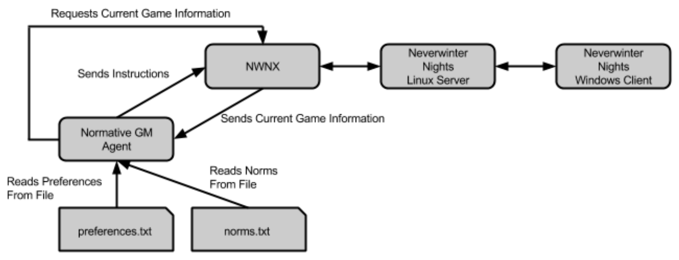
Requests Current Game Information
Sends Instructions
NWNX
Neverwinter Nights Linux Server
Neverwinter Nights Windows Client
Sends Current Game Information
Normative GM Agent
Reads Preferences From File
Reads Norms From File
preferences.txt
norms.txt

Fig. 1: High Level System Architecture

A. Gameplay Preference Specification

Preferences will be used to describe the type of experience the game will present to the player. The intention is that a designer is able to produce a range of different styles of game play with the same GM agent by specifying those styles in terms of these preferences. For example if the designer wishes the game to be an action RPG full of combat, they would describe a preference for plans involving violence.

We define preferences to relate to themes which apply to potential in-game events and gameplay outcomes. In our Neverwinter Nights setting, these themes include: *violence*, *persuasion*, *bribery*, *exploration* and *stealth*. Another dimension is added in terms of the moral implications of the plan, classifying plans involving good-natured player actions as *paragon* and an plans involving evil behaviour as *renegade*.

We define a preference as a specified theme with a numerical weighting out of 10. This definition is a simpler version of the similar concepts in [8], [15] which allow for more detailed specifications such as preferences which need only be satisfied once or until some condition has been met. In contrast, our preference system assumes that the specified preferences are applicable throughout the execution of the game. Additionally the above implementations order preferences based on a numerical value $\mathcal{V}$ for which lower values are more desirable, whereas our system considers higher values of $\mathcal{V}$ preferential over lower values for the purposes of calculations involving probabilities which are described in Section 3.4.

B. Specifying Game Rules As Norms

Norms are used to dictate the GM's behaviour by describing some rules for the game. As developing a generalised normative language is outside the scope of this project, a list of normative options is provided for the user to create their own specifications. These options are defined in an ontology, where each sanction is assigned properties in the same manner as other plans as described in Section III-D, with each property in the figure assumed to have a value 1 unless omitted or stated otherwise. However, the properties with values of -1 represent a distinction from other non-sanction plans and this is elaborated on in the same section.

The norms specified by the user are applicable to the player only. This is because the NPCs in our game scenario are not autonomous and would either not be able to partake in the sanction events, or these events would mostly not be applicable to their scripting. For example, the NPCs are not programmed to navigate dialogue trees between each other therefore they cannot converse in any meaningful way, as some of the sanction events require. They also do not use other abilities such as persuasion and are not able to search for items. Hence the decision was made not to make norms applicable to NPCs as they would be largely unable to undertake them.

C. The Game Master Agent

The GM's core structure is the goal-plan tree, which maps goals to the plans which handle them and plans to their sub-goals. We have added property summary information to the tree and developed algorithms for propagating this information throughout it in a similar fashion to [8]. Finally we developed the reasoning mechanisms themselves so that the GM can reason about preferences, norms and their interactions to attempt to produce desirable gameplay for the player.

The plot points from which the GM can select are contained within the agent's goal-plan tree, concepts of which are described in Section II. Figure 2 shows a subset of the goal plan tree for our GM agent, complete with the themes and actions which are relevant to each plan. Upon execution in Luong's implementation, the main goal is posted and the agent then selects a course of action by selecting plans from the goal-plan tree to satisfy that goal.

In order for our agent to reason about plan selection based on the designer's preferences and norms a property summary is added to each plan in the tree. We follow [8] by adding summary information which details what properties are applicable to the plan. In our case these properties refer to either *actions*, taken from the possible actions in the norm specification ontology required for normative reasoning; or *themes* taken from the possible themes in the preference specification ontology required for preferential reasoning.

The properties applying to plans which trigger an in-game event (as opposed to simply posting further goals) are assigned a values of either 1 or 0. These binary values can be interpreted as true or false. Properties are given a value of 1 if they are applicable to a plan as it is certain that these themes and actions will be present in the resulting gameplay. Values for themes and actions which are not applicable to the plan are omitted from the summary and assumed to be 0 as they will not eventuate as a result of that plan being selected.

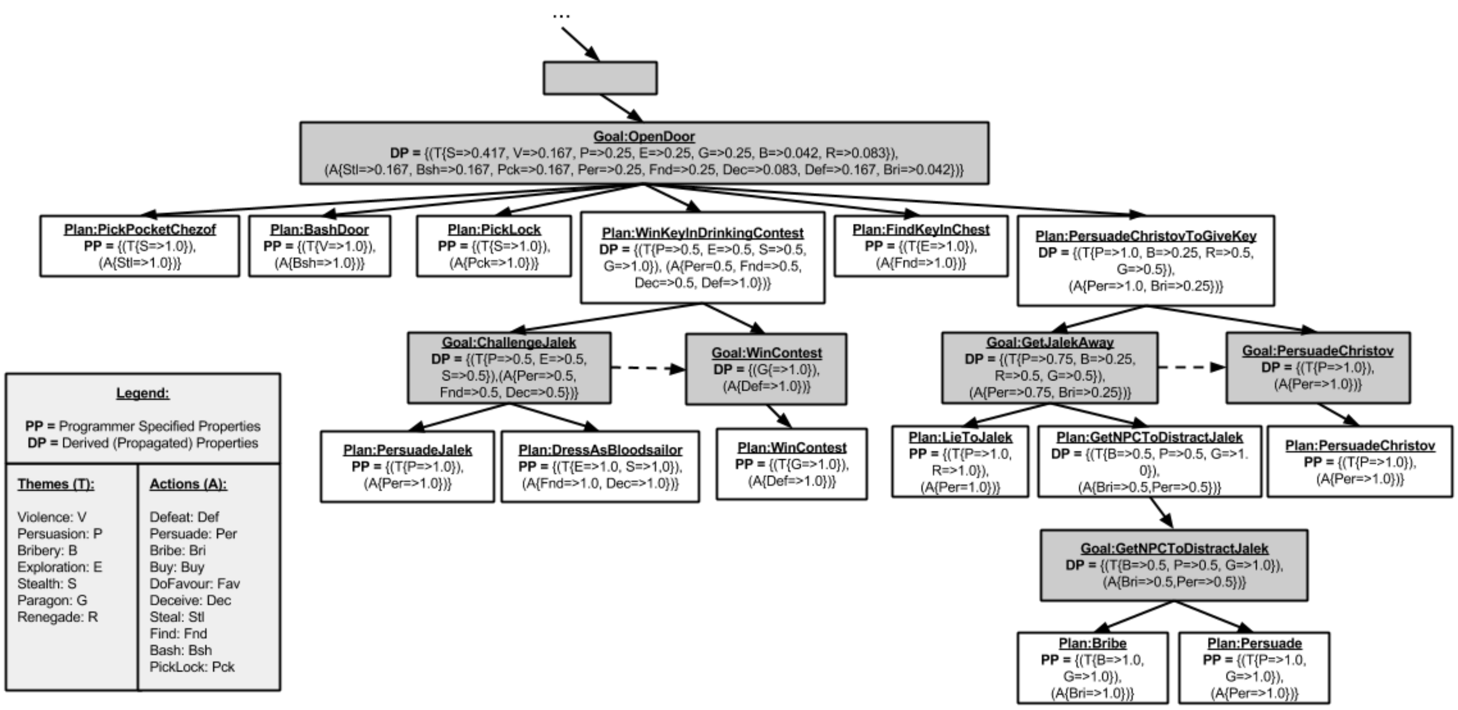
...
Goal:OpenDoor
DP = {(T{S=>0.417, V=>0.167, P=>0.25, E=>0.25, G=>0.25, B=>0.042, R=>0.083}), (A{Stl=>0.167, Bsh=>0.167, Pck=>0.167, Per=>0.25, Fnd=>0.25, Dec=>0.083, Def=>0.167, Bri=>0.042})}
Plan:PickPocketChezof
PP = {(T{S=>1.0}), (A{Stl=>1.0})}
Plan:BashDoor
PP = {(T{V=>1.0}), (A{Bsh=>1.0})}
Plan:PickLock
PP = {(T{S=>1.0}), (A{Pck=>1.0})}
Plan:WinKeyInDrinkingContest
DP = {(T{P=>0.5, E=>0.5, S=>0.5, G=>1.0}), (A{Per=0.5, Fnd=>0.5, Dec=>0.5, Def=>1.0})}
Plan:FindKeyInChest
PP = {(T{E=>1.0}), (A{Fnd=>1.0})}
Plan:PersuadeChristovToGiveKey
DP = {(T{P=>1.0, B=>0.25, R=>0.5, G=>0.5}), (A{Per=>1.0, Bri=>0.25})}
Goal:ChallengeJalek
DP = {(T{P=>0.5, E=>0.5, S=>0.5}),(A{Per=>0.5, Fnd=>0.5, Dec=>0.5})}
Goal:WinContest
DP = {(G{=>1.0}), (A{Def=>1.0})}
Goal:GetJalekAway
DP = {(T{P=>0.75, B=>0.25, R=>0.5, G=>0.5}), (A{Per=>0.75, Bri=>0.25})}
Goal:PersuadeChristov
DP = {(T{P=>1.0}), (A{Per=>1.0})}
Legend:
PP = Programmer Specified Properties
DP = Derived (Propagated) Properties
Themes (T):
Violence: V
Persuasion: P
Bribery: B
Exploration: E
Stealth: S
Paragon: G
Renegade: R
Actions (A):
Defeat: Def
Persuade: Per
Bribe: Bri
Buy: Buy
DoFavour: Fav
Deceive: Dec
Steal: Stl
Find: Fnd
Bash: Bsh
PickLock: Pck
Plan:PersuadeJalek
PP = {(T{P=>1.0}), (A{Per=>1.0})}
Plan:DressAsBloodsailor
PP = {(T{E=>1.0, S=>1,0}), (A{Fnd=>1.0, Dec=>1.0})}
Plan:WinContest
PP = {(T{G=>1.0}), (A{Def=>1.0})}
Plan:LieToJalek
PP = {(T{P=>1.0, R=>1.0}), (A{Per=1.0})}
Plan:GetNPCToDistractJalek
DP = {(T{B=>0.5, P=>0.5, G=>1. 0}), (A{Bri=>0.5,Per=>0.5})}
Plan:PersuadeChristov
PP = {(T{P=>1.0}), (A{Per=>1.0})}
Goal:GetNPCToDistractJalek
DP = {(T{B=>0.5, P=>0.5, G=>1.0}), (A{Bri=>0.5,Per=>0.5})}
Plan:Bribe
PP = {(T{B=>1.0, G=>1.0}), (A{Bri=>1.0})}
Plan:Persuade
PP = {(T{P=>1.0, G=>1.0}), (A{Per=>1.0})}

Fig. 2: A Subtree of the GM's Overall Goal-Plan Tree with Propagated Property Summary Information

However, the agent must be able to make a decision at all levels of the tree, even if the plans it has to select from do not directly bring about in-game events. In these cases, properties are not directly applicable to a plan but may be applicable to plans at a lower level of the sub-tree which extends from that plan. Thus the agent must be given some information about the plans in the levels of that sub-tree in order to perform reasoning about which plan to select at a higher level.

To this end we extend the previous work of [8] by assigning a probability to the property's value. This value is the probability that, if the plan is selected, at least one lower level plan from the sub-tree will be subsequently selected which triggers an in-game event that satisfies the property. For example, in Figure 2 the plan *GetNPCToDistractJalek* posts a sub-goal with the same name. There are two plans which handle this sub-goal, one of which involves having the player use the persuasion ability. At least one plan must be selected to handle the sub-goal, therefore if *GetNPCToDistractJalek* is selected, there is a 50% chance that the ensuing gameplay will involve persuasion. Hence the *GetNPCToDistractJalek* plan is assigned the value of 0.5 for the *persuasion* property.

Note that this probability value is a static calculation which is determined by hand and assigned to variables during implementation. It does not take into account preferences, as the actual probability of a given plan being selected would depend on the norms and preferences specified. However we have used these probability values because they are a clear way to provide information to the agent so that it can "look ahead" at the implications of selecting any given plan.

D. Normative and Preferential Reasoning

Plan summary information is calculated for each goal and plan by recursively propagating values from the leaf nodes upwards through the tree. As stated above, the calculation results in a probability that properties will apply to in-game outcomes, however this calculation is done differently for each node depending on if the node is a goal or a plan. This is due to the way the two relate to each other as explored by [8], namely the AND/OR relationships described in Section II.

Propagation to goal nodes is achieved by averaging a property's child node probabilities for each possible property. We define the probability of a theme t occurring for goal g as $\Pr(g_t)$ in Formula 1, where $children(g)$ is a set returning the child nodes of g and $\Pr(i_t)$ is the probability of t occurring in a child node i.

$$\Pr(g_t) = \sum\nolimits_{i \in children(g)} \frac{\Pr(i_t)}{|children(g)|} \tag{1}$$

For example in Figure 2, the theme probabilities for the goal *GetNPCToDistractJalek* are calculated as follows:

- $Bribery$: $1.0/2 + 0.0/2 = 0.5$
- $Persuasion$: $0.0/2 + 1.0/2 = 0.5$

Here we see that there is a 50% chance of the goal *GetNPCToDistractJalek* triggering gameplay involving bribery as there are two plans which handle it, one of which involves bribery (likewise for persuasion). Yet there is a 100% chance that the goal will trigger gameplay involving the paragon theme as both plans which achieve that goal carry that theme.

Propagation to a plan node is more involved as it must account for AND connections between goals when calculating probabilities. Here we must consider the probability formula for non-mutually exclusive events as detailed in [23], [24], which is the complement of the product of the complements of the probabilities. We define the probability of a theme t occuring in a plan l as $\Pr(l_t)$ in Formula 2, where $children(l)$ is a set returning the child nodes of l and $\Pr(i_t)$ is the probability of t occurring in a child node i:

$$\Pr(l_t) = 1 - \prod_{i \in children(l)} (1 - \Pr(i_t)) \quad (2)$$

For example in Figure 2 the action probabilities for the plan *PersuadeChristovToGiveKey* are calculated as follows:

- $Persuasion$: $1 - (1 - 0.75)(1 - 1.0) = 1.0$
- $Bribery$: $1 - (1 - 0.25)(1 - 0.0) = 0.25$

Persuasion has a 100% chance of resulting from the plan *PersuadeChristovToGiveKey* being selected as, regardless of how the goal *GetJalekAway* is handled, the plan *PersuadeChristov* must also be achieved, which will definitely result in gameplay involving persuasion. Yet, there is only a 25% chance that bribery will occur as the goal *PersuadeChristov* does not involve this action and thus the 0.25 probability of it occuring as a result of handling the goal *GetJalekAway* remains.

Sanction Properties: Plans in the goal-plan tree which result in in-game events are assigned a probability of either 0 or 1. However, plans which are executed as the result of a sanction may have another possible value, -1. Instead of being a probability, this value indicates that, while no themes are applicable to the gameplay outcomes of the plan, the outcomes are detrimental to the player's ability to engage in behaviour.

For example the *WeaponConfiscated* sanction leaves the player unarmed, severely impeding their capacity for violent actions. Therefore this sanction is assigned the *violence* property with a value of -1, allowing the agent to reason about the implications of the player to triggering this sanction on any further plans involving violence.

Evaluating the Plan Score: When the GM agent is executed, the root goal of the goal plan-tree is posted and the agent begins traversing this tree – executing plans to address goals and posting their sub-goals in turn. For each goal that is posted, the GM evaluates the applicable plans which handle the goal and calculates a score for each of them. This score is calculated based on: the preferences specified; the inherent plan properties; and the properties of any norms which may be violated by the player in carrying out the plan.

First the GM checks the actions which the plan entails against the set of user specified norms. If any of these actions match an action which is *obliged* or *prohibited* through a specified norm, then that norm is considered to be *applicable*. For each applicable norm, the themes which are associated with the sanction are added to the set of the themes for the plan. These probabilities are then aggregated to calculate the probabilities for themes which appear more than once, again using the complement of the product of complements as we use for propagating plan summary information.

Once the set of plan themes has been aggregated, the GM: (i) compares this set to the preferences specified by the user; (ii) finds the intersection of themes which appear both in the specified preferences and the set of plan themes, and for each theme a value is calculated (product of preference weight and probability of that theme in the plan summary information); and (iii) sums the values for all themes in the intersection to produce a score for the plan. We define the score for a plan l as S_l in Formula 3, where: r is the set of specified preferences; $themes(x)$ the set of themes relevant to a plan or a set of preferences x; $weight(r_i)$ the weight for theme i as specified within a set of preferences; and $\Pr(l_i)$ the probability of a theme i being present in the gameplay executed by plan l.

$$S_l = \sum_{i \in (themes(r) \bigcap themes(l))} weight(r_i) \times \Pr(l_i) \quad (3)$$

The GM ignores themes not present in the intersection of preference and plan themes. This is because themes not specified as preferences are assumed to have a preference weight of 0 and similarly themes not appearing in the plan properties are assumed to have a probability of 0. In both cases the theme would contribute a value of 0 to the overall score. Once the scores for all applicable plans have been calculated the GM executes the plan with the highest score, applying any in-game effects and posting the plan's sub-goals if it has any.

IV. Evaluation

In this section we detail the user experiment undertaken and an analysis of the results. A case study was also undertaken to illustrate the inner workings of the new GM. While it has been omitted here due to space constraints, full details of the case study can be sighted online[1]. Our experiment involved a mix of 10 male and female participants which had a background in video game design. Given that access to professional video game designers is difficult to achieve, our recruitment focused mainly on game design program graduates. Our final participant pool comprised of 8 recent graduates (2 of which are currently working at game development studios) and 2 game designers who are currently in the industry. No subjects had played the game Neverwinter Nights prior to the experiment so some brief instructions were given around the controls and objectives of the game scenario.

Our experiment involved 2 systems: System A and System B. System A was the original GM by Luong et al. [2] on which our system was based while System B was the normative GM which we developed.

Each participant was given an explanation of the system and given a printed copy of the system's ontology and a detailed explanation of how the language is used to specify gameplay preferences and norms. The participant was then asked to think of a style of game they would like to create and then attempt to describe it to the system using the language described in the ontology. The participants were directed to specify 2 or more preferences and 2 or more norms. The participant then played through System A and System B. Users were not told which system was which and half of the experiments presented System A first, whilst the other half began with System B. Once the participant had tested both systems they were given the opportunity to do additional runs through either system and to specify new preferences and norms if they so wished. When the participant had finished their testing session they were given a questionnaire to fill out.

When specifying norms and preferences, participants were encouraged to attempt to create gameplay situations in which norms would be broken or exploited by the GM. This was so that the designers were more likely to observe the norm enforcement mechanics, as there was a good chance these features may not have been evident during the gameplay testing if the designers had not taken this into consideration.

[1]The full case study is available at http://goo.gl/U777M1

Whilst this experimental setup is similar to that of a "player testing" session, we draw an important distinction. The play-through of the game itself is not intended to be an evaluation of the gameplay from a player's point of view, it is intended to be a test run from the designer's point of view. The questionnaire is reflective of this – its questions being around how well the GM conforms to their specifications as opposed to how fun the game is or how much "replay value" it has. Our experiment is designed to be a shortened and simplified development iteration loop comprising design, implementation and testing phases and is intended to be reflective of the video game development process.

The Questionnaire: given to the participants contained both quantitative and qualitative questions:

1) Which system produced the type of gameplay you described in your preferences? *Answered on a 5 point Likert scale labelled "1-Definitely System A" to "5-Definitely System B"*
2) Which system best enforced the norms you defined? *Answered as (1).*
3) Which system best produced the type of narrative you were trying to achieve with the preferences and norms you specified? *Answered as (1).*
4) With regards to the system which performed better in the above 3 aspects, how well did the system satisfy the preferences and enforce the norms you defined? *Answered on a 5 point Likert scale labelled "1-Only Slightly" to "5-Very Well"*
5) Given minimal training and experience with the system, how easy are the preference and norm definition languages to read and use? *Answered on a 5 point Likert scale labelled "1-Very Difficult" to "5-Very Easy"*
6) How suitable do you think this approach of specifying gameplay via preferences and norms would be to designing video games with flexible narrative? *Answered on a 5 point Likert scale labelled "1-Not Suitable At All" to "5-Highly Suitable"*
7) What are the positive aspects of the system, if any? *Answered in a text box which accepted a paragraph*
8) What aspects of the system are in need of improvement, if any? *Answered in a text box which accepted a paragraph*

Analysis of Results: The results show that participants identified the difference between the two systems and that they believed the normative GM was doing a better job of satisfying their preferences, enforcing their specified norms and overall producing the type of gameplay they desired than the original GM. They also show that the participants felt that the system performed well in these respects individually and was a suitable method of designing games with flexible narrative. These results are summarised in Figure 3.

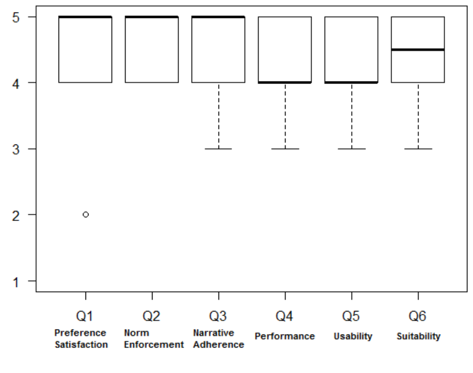

Fig. 3: Box plot detailing the spread of results.

We analysed the results with the Wilcoxon Signed-Rank Test to test for significance as we had only a single ordinal dependant variable (user score) and only a single sample in each case. This test is also relevant because it does not assume data is normally distributed. The result for each question was determined to be statistically significant (scores below 0.05).

Questions 7 and 8 were requesting qualitative feedback on the strengths and weaknesses of the system. Overall the positive comments mostly dealt with the potential of the system and how easy it was actually specify the norms and preferences. Users also identified some areas in need of improvement as the learning curve for the norm and preference specification languages; some of the terminology used in the ontology; and the fact that the GM restricts some of the player's agency by specifically telling them what to do.

Feedback from participants was encouraging in that they saw potential in the system's functionality. However, users also expressed some views that the system was initially somewhat challenging to understand and this will be addressed in future work. Some learning curve is to be expected however, and we contend that a single 30 minute evaluation of the system is not sufficient for a user to become completely comfortable with it. Users also expressed some disappointment that the GM specifically directed them as to what players should do within the game. This is inherent in the original GM, however as the scenario which the GM can handle becomes larger, the potential increases for the GM to direct entire plot lines at a higher level, rather than directing the player in atomic actions. This could also lead to interesting future work.

V. Conclusion

Flexible narrative in video games is a well investigated problem to which a number of solutions have been proposed. We have built upon one such solution [2] which uses an automated intelligent BDI agent to construct a narrative for the player. In this paper we have proposed and implemented a preference and norm specification language for the purposes of allowing game designers to create games with flexible narrative using an automated intelligent game master. The system is designed to be human readable and accessible to developers who have little knowledge of the implementation details of the GM itself.

The results from the user experiment showed that users found the system to do a good job at producing the type of gameplay they desired by satisfying their specified preferences and enforcing their norms. The results also show that users found the system to be fairly easy to use with minimal instruction and that they believe the system to be suitable for developing games with flexible narrative.

Other implementations of preference [13], [15] and norm [6] specification languages are more expressive, allowing for more involved and complicated definitions. Our work focused on human readability and thus implemented an intentionally simplified system of preference and norm specification designed to be easily learnt and understood by users. Whilst more complex definition languages are outside the scope of this work, a larger and more expressive ontology may also make it more difficult for users to comprehend the system and learn how to use it. Future work may include attempting to expand the expressiveness of the languages whilst trying to maintain a focus on usability for game designers.

References

[1] Wizards RPG Team, *Dungeons & Dragons Dungeon Master's Guide: Roleplaying Game Core Rules, 4th Edition*. Wizards of the Coast, 2008.

[2] B. V. Luong, J. Thangarajah, F. Zambetta, and M. Hasan, "A BDI game master agent for computer role-playing games," in *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 1187–1188.

[3] BioWare, "Neverwinter Nights," CD-ROM computer role-playing game, 2002.

[4] A. S. Rao and M. P. Georgeff, "Modeling Rational Agents Within A BDI-Architecture," in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, Knowledge Representation and Reasoning Inc. Morgan Kaufmann, 1991, pp. 473–484.

[5] A. S. Rao, M. P. Georgeff *et al.*, "BDI Agents: From Theory to Practice," in *International Conference on Multi-Agent Systems*, vol. 95, 1995, pp. 312–319.

[6] N. Alechina, M. Dastani, and B. Logan, "Programming Norm-aware Agents," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems (AAMAS), 2012, pp. 1057–1064.

[7] D. Dybalova, B. Testerink, M. Dastani, B. Logan, F. Dignum, and A. Chopra, "A Framework For Programming Norm-Aware Multi-Agent Systems," in *Proceedings of the 15th International Workshop on Coordination, Organisations, Institutions and Norms*. International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2013.

[8] S. Visser, J. Thangarajah, and J. Harland, "Reasoning About Preferences in Intelligent Agent Systems," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, ser. International Joint Conferences on Artificial Intelligence (IJCAI'11). Association for the Advancement of Artificial Intelligence Press, 2011, pp. 426–431.

[9] M. E. Bratman, D. J. Israel, and M. E. Pollack, "Plans and resource-bounded practical reasoning," *Computational intelligence*, vol. 4, no. 3, pp. 349–355, 1988.

[10] L. Padgham and M. Winikoff, *Developing intelligent agent systems: A practical guide*. John Wiley & Sons, 2005, vol. 13.

[11] L. Padgham, J. Thangarajah, and M. Winikoff, "Tool support for agent development using the prometheus methodology," in *Quality Software, 2005.(QSIC 2005). Fifth International Conference on*. IEEE, 2005, pp. 383–388.

[12] M. Winikoff, "JACK Intelligent Agents: An Industrial Strength Platform," in *Multi-Agent Programming*. Springer, 2005, pp. 175–193.

[13] T. C. Son and E. Pontelli, "Planning with preferences using logic programming," in *Logic Programming and Nonmonotonic Reasoning*. Springer, 2004, pp. 247–260.

[14] V. Lifschitz, "Answer set programming and plan generation," *Artificial Intelligence*, vol. 138, no. 1, pp. 39–54, 2002.

[15] M. Bienvenu, C. Fritz, and S. A. McIlraith, "Planning with qualitative temporal preferences." *Principles of Knowledge Representation and Reasoning (KR'06)*, vol. 6, pp. 134–144, 2006.

[16] J. F. Hübner, J. S. Sichman, and O. Boissier, "$\mathcal{S}-\mathcal{M}$OISE$^+$: A Middleware for Developing Organised Multi-agent Systems," in *Coordination, Organizations, Institutions, And Norms In Multi-Agent Systems*. Springer, 2006, pp. 64–77.

[17] M. Esteva, D. De La Cruz, and C. Sierra, "ISLANDER: An Electronic Institutions Editor," in *Proceedings Of The First International Joint Conference On Autonomous Agents And Multiagent Systems: Part 3*. Association for Computing Machinery, 2002, pp. 1045–1052.

[18] M. Esteva, B. Rosell, J. A. Rodriguez-Aguilar, and J. L. Arcos, "Ameli: An Agent-Based Middleware For Electronic Institutions," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*. Association for Computer Machinery, 2004, pp. 236–243.

[19] M. Esteva, J. Padget, and C. Sierra, "Formalizing A Language For Institutions And Norms," in *Intelligent Agents VIII*. Springer, 2002, pp. 348–366.

[20] J. F. Hübner, J. S. Sichman, and O. Boissier, "$\mathcal{M}$oise$^+$: towards a structural, functional, and deontic model for mas organization," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. ACM, 2002, pp. 501–502.

[21] D. Thue, V. Bulitko, and M. Spetch, "Passage: A demonstration of player modeling in interactive storytelling." in *AIIDE'08*, 2008, pp. 227–228.

[22] M. Trenton, D. Szafron, J. Friesen, and C. Onuczko, "Quest patterns for story-based computer games." in *AIIDE'10*, 2010.

[23] J. Devore, *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 2011.

[24] Y. S. Chow and H. Teicher, *Probability theory: independence, interchangeability, martingales*. Springer, 2003.