AI Authoring for Virtual Characters in Conflict

Paulo Gomes and **Arnav Jhala**
UC Santa Cruz, CA, 95064, USA

Abstract

Game developers strive to have engaging believable characters in their work. One of the elements that has been pointed out as contributing to believability is social behavior. A category of social behavior is interpersonal conflict. In our current research we compare two AI approaches to model NPC conflict resolution strategies: one using the reactive planning language ABL and another using the AI framework FAtiMA. We identify the following metrics to evaluate social behavior modeling: mapping theory, emotion, model checking, variability, policy change. In our analysis we found it was easier to map conflict concepts in ABL and the model checking process was faster. FAtiMA had better support for emotion and other emergent attributes.

Introduction

Game developers strive to have engaging believable characters in their work. One of the elements that has been pointed out as contributing to believability is social behavior (Mateas 1999). Although character's social behavior is often represented in games through interactive dialogs and cinematics, few titles consider it as a core game mechanic (Prom Week being one of the exceptions (McCoy et al. 2010)). Furthermore, commonly used AI techniques such as dialog trees and behavior trees seem to lack the support that social play may require. Namely, dialog trees can lead the story thread to break down due to unforeseen inconsistencies (Wardrip-Fruin 2012). Consequently, developing high-level behavior authoring languages and frameworks for video game social behavior is an important avenue in research (Yannakakis 2012).

Among social behavior mechanisms in contemporary video game titles, it is becoming popular to have the main character's actions valued with one of two opposing values (Light vs Dark, Renegade vs Paragon). These actions control the main character's position in a scalar scale. Moreover, this position determines what type of actions the player is allowed to perform with different characters (e.g. if the player has thwarted the Empire's missions she might find herself unwelcome at the Sith Lords' kickback).

However, a system like this may not capture the nuances of the interpersonal character history a designer desires. Consider that as an expressive goal, if a game character has recently shot an NPC sidekick (possibly by accident) this NPC should be more reluctant to cooperate with the player. Additionally, consider that as a player you may need to convince the said NPC to perform a dangerous task. When the player performs a request to the NPC a conflict situation arises: two characters with conflicting goals (NPC staying alive, Main character saving the kingdom). This conflict should be performed in a way that considers the expressive contextual requirement (taking into account the accidental shot).

Using the mentioned opposing sides mechanism will hardly map the contextual condition. One could resort to a flag system, having a flag for the NPC being upset with the main character and setting it to true when the shot happens. However, such flags and handlers exponentially increase the authoring burden in specification of the branching narrative leading to tightly controlled story arcs and its negative effect on player agency. Frameworks that model social behavior explicitly may provide a better fit for the described conflict constructs. Additionally, they may support different degrees of cooperation and take into account authorable character traits. These traits would allow to maintain interaction consistency even if the player had several side-kicks to choose from.

Many AI approaches are being explored for modeling intelligent virtual characters that could provide a rich set of believable interactions. Behavior trees for instance tend to be used as a more centralized approach, that is, a reasoning algorithm that operates on shared memory and behavior libraries for all characters. On the other hand there are approaches that focus more on individual agents. We will look into more character directed approaches. Specifically we will analyze conflict situations as an example of rich social interactions.

To support these situations the modeling systems should be robust and expressive, take into account context, while being finely tunable by designers. In our current research we compare two AI approaches to model NPC conflict resolution strategies: one using the reactive planning language ABL (Mateas and Stern 2004), and another using the AI framework FAtiMA (Dias, Mascarenhas, and Paiva 2011).

In our work we wanted to study the extent to which these approaches supported conflict expressiveness but also to layout a methodology on how to evaluate a system's expressiveness regarding a social behavior feature (in our case conflict).

In this article we will start by explaining what we understand by interpersonal conflict. We will continue by giving a brief description of the two frameworks considered. Following this we look at a concrete conflict scenario and how it can be modeled in both. Afterwards, based on the modeling process described, we compare how both frameworks are able to support the interpersonal conflict context. Finally, we summarize the comparison and point possible avenues of future work.

Definitions

In the current work we are trying to model interpersonal conflict as a social behavior feature example. For clarification, by interpersonal conflict we mean:

> a situation in which two individuals have opposing interests and at least one of them acknowledges said interests.

A system that models interpersonal conflict should encode different scenarios and different ways of dealing with them. Consequently, it is also relevant to define a categorization of conflict resolution strategies. In a initial approach, we decided to consider a simple and clear categorization that would still allow for some variability. The one presented is inspired in the summarizing categorization proposed in (Rahim 2010). The categories are the following:

- **Dominating**: the individual pursues personal goals with low concern for the interests of the other individual. For instance, consider that a manager is contacted by a subordinate that tells him that he probably won't be able to meet a defined deadline. A dominating strategy by the manager might be to threaten to fire him if he does not meet the deadline.
- **Integrating**: the individual tries to account for personal but also other party's interests. In the previously described scenario, the manager might suggest a bonus in compensation for over time, and at the same time try to discuss with the subordinate the underlying reasons for the problem.
- **Avoiding**: the individual does not address either of the goals. In the same scenario, the manager might tell the subordinate to come back later because he is currently too busy.
- **Accommodating**: the individual thwarts personal goals and addresses other party's goals. In our scenario, the manager could simply tell the subordinate that it is fine that the deadline is missed without looking into the reasons for the schedule slide.

Rahim also proposed a Compromise category (Rahim 2010) as intermediate compared to the other four. In an initial modeling effort we considered that it would be more valuable to focus on the presented four strategy types with clearer bounds. One can see a representation of the different strategies in Figure 1.

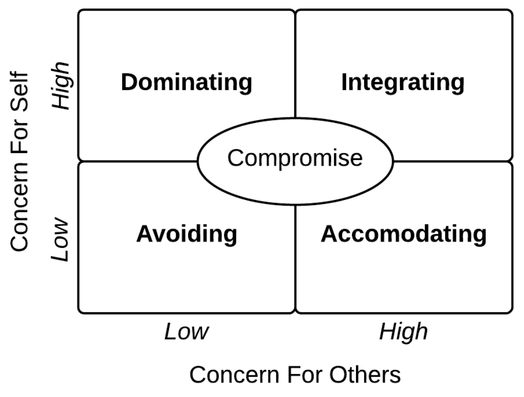

Figure 1: Conflict resolution strategy types.

Modeling approaches

The two interpersonal modeling approaches studied were ABL and FAtiMA. Bellow we give a quick introduction to both, just enough to understand how they can be used to perform the mentioned modeling.

FAtiMA

Fearnot AffecTIve Mind Architecture (Dias, Mascarenhas, and Paiva 2011) is a computational model of appraisal theory of human emotions which is authorable in XML. Generically appraisal theory claims that emotions are valanced interpretations of perceived events. Appraisal theories also claim that these interpretations depend on several appraisal variables: desirability of an event, praiseworthiness of an action, likability attitude towards an object, likelihood of a future event, among other. FAtiMA models this appraisal derivation process. Furthermore, it generates an emotional state based on these appraisal variables using an OCC theory of emotions (Ortony, Clore, and Collins 1990) inspired process (affect derivation). A strong motivation for having an agent architecture with a model of emotion is not only that it can simulate the emotional processes, but also, that emotion has been shown to be an integral process of human's decision making process (Damasio 1994).

An important consideration in modeling interpersonal conflict in FAtiMA is that individual agents' behavior is defined by a STRIPS-like planner (Fikes and Nilsson 1972). Core to this planner are actions and goals. Actions have preconditions and effects. For instance, eating an apple might have as precondition that the agent believes it is holding an apple and as an effect that the apple is eaten. Goals have success conditions. For example, the agent Adam might have as a goal nourishment, with the success condition that it has recently eaten food. At any time point an agent has several potential goals to pursue in memory. The goal selected will depend on its relative importance. Importance is calculated according to the associated appraisal variables. For instance, if two goals are desirable, the one with prospective higher desirability will be selected.

FAtiMA architecture has actually been used to model conflict (Campos, Martinho, and Paiva 2013). However, in the referenced work there is a stronger emphasis on the conflict detecting process rather than in the strategy choice phase.

ABL

ABL is a programming language for reactive planning (Mateas and Stern 2004). It was designed to support virtual characters and it compiles to Java. It follows the principles of a hierarchical task network style planner. Goals are not defined by preconditions and success conditions. Instead, they are defined by the possible solutions for that goal. These solutions are called behaviors. Behaviors are recipes to reach the goal. A behavior itself has preconditions and children. These children can be atomic actions or new goals. Executing the behavior is performed by executing its children. At any time the system maintains a tree of the current active behaviors.

Like FAtiMA, ABL has atomic actions called acts. They can be declared actions in the world or mental acts. Mental acts are arbitrary pieces of Java code. Another crucial structure of the language is the Working Memory Element (WME). A WME is a piece of information representing a belief of the AI about the current state of the world or of a previous event. WMEs are the fundamental element of information representation in ABL. They can be referenced in preconditions. A behavior is only activated if there is a consistent unification of all the variables in the preconditions. Additionally, WMEs can be changed in mental acts. A more detailed decription of the architecture is available at http://abl.soe.ucsc.edu.

Conflict Modeling

Now that we briefly looked at the two frameworks, we will discuss how they can be used to model interpersonal conflict. We will start by describing a scenario that will serve as an example for both. Consider the following example scenario:

> Adam asked Bob to buy him a ticket so they could both go to a show. As Bob is friends with Adam, he bought it. Afterwards, Adam told Bob he did not want to go to the concert anymore so would not pay the ticket back. Bob told Adam that he should pay no matter what.

In the following scenario the characters have opposing interests: Bob wants to get his money back, and Adam doesn't want to run out of money. They both realize there is a conflict of interests when Adam tells Bob he is not going to pay. There are acknowledged opposing interests between two individuals, thus according to our definition this is an interpersonal conflict. This scenario is inspired in a conflict description gathered through crowd-sourcing (Swanson and Jhala 2012). Additionally, we may notice that Bob tries to ensure his goal is achieved with little concern for Adam's interests. Consequently, Bob's strategy can be classified as dominating. In the remainder of this section we will focus on modeling the moment in which Bob has to decide how to respond to Adam's refusal to pay.

ABL Modeling

The just presented scenario can be described in ABL. Although ABL supports the synchronized behavior of individual virtual characters, we decided to use the conflict scenario as an abstraction rather than individual agents. Consequently, one ABL Entity models the whole conflict situation. This approach has been used in practice by the Games and Playable Media Group in the IMMERSE project (IMMERSE 2012).

Keeping in mind that we are using the situation as an abstraction, and not the agents, we define the following types of goals: *infer conflict* and *resolve conflict*. There is only one infer conflict goal that is used to the detect conflicts. There are different corresponding behaviors for this goal that represent different conflict situations (e.g. owing conflict). The behavior selected depends on the preconditions that encode the conflict context (e.g. a character owes another money). On the other hand, there are several resolve conflict goals which represent trying to resolve different types of conflict situations (e.g. resolveChoresConflict). Furthermore, each resolve conflict goal has different behaviors corresponding to different resolution strategies (dominating, avoiding, accommodating, integrating).

Consider the following infer conflict behavior example and focus for now on the preconditions.

```
// Infer owing conflict
sequential behavior inferConflict(){
    precondition {
        (CancelOwesWME characterInDebt::characterInDebt
         characterInDebtTo::characterInDebtTo)
        (OwesWME characterInDebt==characterInDebt
         characterInDebtTo==characterInDebtTo)
        !(ConflictWME type==ConflictType.OWE
         characterA==characterInDebt
         characterB==characterInDebtTo)
    }
    specificity 2;
    mental_act{
        RelWME upsetWME;
        upsetWME = new RelWME("upset",
         characterInDebtTo,
         characterInDebt);
        ConflictWME conflictWME = new ConflictWME(ConflictType.OWE,
         characterInDebt,
         characterInDebtTo);
        BehavingEntity.getBehavingEntity().addWME(upsetWME);
        BehavingEntity.getBehavingEntity().addWME(conflictWME);
    }
    with (priority 4) subgoal resolveOwesConflict();
}
```

This behavior tries to encode the type of conflict described in the example scenario. It detects if a character is in debt to another and refuses to pay through a CancelOwesWME condition:

```
(CancelOwesWME characterInDebt::characterInDebt
characterInDebtTo::characterInDebtTo)
```

The behavior has two children that will be executed sequentially (notice the sequential tag before the name). These children are: a mental act, in which several WMEs are added (recording what happened); a subgoal of resolving this specific type of onflict (resolveOwesConflict). Now consider one of the possible behaviors corresponding to the resolveOwesConflict:

```
// Dominating strategy
sequential behavior resolveOwesConflict(){
    precondition {
       (ConflictWME type==ConflictType.OWE characterA::characterA
        characterB::characterB)
       (PersonalityWME character==characterB
        type==PersonalityType.HIGH_CONCERN_SELF)
    }
    specificity 2;
    mental_act{
       AblCompilation.getInstance().
        println(characterB+": "+"You better find a way to pay me!");
```

```
      ConflictActionWME conflictActionWME;
      conflictActionWME = new ConflictActionWME(ConflictType.OWE,
        ResponseType.DOMINATING,
        characterB,characterA);
      BehavingEntity.getBehavingEntity().addWME(conflictActionWME);
    }
}
```

The behavior will only be selected if the character has high concern for its personal interests:

```
(PersonalityWME character==characterB
type==PersonalityType.HIGH_CONCERN_SELF)
```

This condition is inspired in the presented definition of dominating strategy. The expression of the strategy results in the lending character to express his goal with no interest for the goals of the other character ("You better find a way to pay me!"). This was what happened in our example scenario. However, it is easy to define alternative strategies. It is only necessary to add a new resolveOwesConflict behavior. For instance, one can define an integrating behavior with the same preconditions, but in which the lending character suggests payment over time. Since the preconditions are the same, this will cause the generated system to choose randomly between the two, causing variability. Alternatively, an accommodating behavior could be defined for when the lending character has low concern for self (precondition) resulting in him giving up getting his money back and expressing it.

FAtiMA Modeling

The same alternative strategies can be modeled in FAtiMA. Contrary to ABL, using the situation as an abstraction is not an option in FAtiMA since an individual emotional state is maintained per agent and it affects the decision making process.

In ABL we define that each goal corresponds to a different resolution strategy. Instead of selecting the strategy through preconditions, the strategy will be chosen according to the different importance the agent gives to the goals: goals with higher importance for an agent are more likely to be pursued than those with lower importance. This importance corresponds to a desirability that in turn will affect the agent's emotional state.

Moreover, since goals do not have recipes on how to execute them (behaviors in ABL) the atomic actions will have to have a strategy bias, meaning that if a strategy is chosen, certain actions are more likely to be selected for execution. This is achieved by setting the value of a character attribute if an action is chosen, and using that same attribute/value pair in a goal success condition. Consider the two following example actions:

```
<Action name="UnilateralDemand([person],[objectOwed])">
<PreConditions>
    <Property name="[person](isPerson)" operator="=" value="True" />
</PreConditions>
<Effects>
<Effect probability="1.0">
   <Property name="[AGENT](resolvedOwesConflict)"
    operator="=" value="True" />

<Property operator="=" name="[AGENT](isRude)" value="True"/>
</Effect>
</Effects>
</Action>

<Action name="FallbackDemand([person],[objectOwed])">
<PreConditions>
    <Property name="[person](isPerson)" operator="=" value="True" />
</PreConditions>
<Effects>
<Effect probability="1.0">
   <Property name="[AGENT](resolvedOwesConflict)"
    operator="=" value="True" />
<Property operator="=" name="[AGENT](lostMoney)" value="True"/>
</Effect>
</Effects>
</Action>
```

Both UnilateralDemand and FallbackDemand will resolve the owes conflict:

```
...
<Effect probability="1.0">
   <Property name="[AGENT](resolvedOwesConflict)"
...
```

However, the first will result in the character being rude and the second in losing money:

```
// UnilateralDemand
...
<Effect probability="1.0">
<Property operator="=" name="[AGENT](isRude)" value="True"/>
...

// FallbackDemand
...
<Effect probability="1.0">
<Property operator="=" name="[AGENT](lostMoney)" value="True"/>
...
```

In turn, the agent can have two interest goals: of maintaining calm by not being rude, and not losing money. Consider that the first will be violated by UnilateralDemand and the second by FallbackDemand. Additionally, consider a third goal that has as a precondition that someone owes the agent and as a success condition that the corresponding conflict is resolved:

```
<ActivePursuitGoal
name="ResolveOwesConflict([character],[objectOwed])">
<PreConditions>
<Property name="[character](ownsTo)"
    operator="=" value="[SELF]" />
</PreConditions>
<SuccessConditions>
<Property name="[SELF](resolvedOwesConflict)"
operator="=" value="True" />
</SuccessConditions>
<FailureConditions>
</FailureConditions>
</ActivePursuitGoal>
```

We again have the example scenario modeled. The element that is missing is the strategy selection according to the goal importance. That is defined in the following way:

```
<Goals>
<Goal name="ResolveOwesConflict([character],[objectOwed])"
   importanceOfSuccess="6" importanceOfFailure="6" />
<Goal name="MaintainCalm()"
   importanceOfSuccess="3" importanceOfFailure="3" />
<Goal name="MaintainMoney()"
   importanceOfSuccess="4" importanceOfFailure="4" />
</Goals>
```

In this case the most important goal is resolving the conflict, followed by not losing money, and keeping calm lastly. Not considering other potential goals, the described character, when having someone owe it money, will try to do a unilateral demand. A dominant strategy is favored. If MaintainCalm was more important than MaintainMoney, but still less important than ResolveOwesConflict, the character would end up choosing to fallback on its demands. An accommodating strategy would be favored. Finally, if ResolveOwesConflict was less important than both, the agent would chose neither, resulting in a avoiding strategy. In the three cases, since both MaintainMoney and MaintainCalm goals are activated, and the agent considers actions that would violate them, the agent worries about them. This is implemented as a fear emotion added to the characters emotional state.

Modeling comparison

We have already started to identify differences in both modeling strategies as we were describing them. Here we will go into greater depth regarding the authorial differences between frameworks. We assume that when the system is authored the designers have narrative elements in mind (character traits, action tendencies, desired outcomes, general theme, etc.). Consequently, it is valuable to evaluate the methods in a narrative perspective. We will consider the approach described in (Thue, Bulitko, and Spetch 2008) for categorizing design decisions in interactive narratives. That is not to say that the output of both systems will necessarily be a story. It might in fact score low in Ryan's metrics for classical narrativeness (Ryan 2006).

In (Thue, Bulitko, and Spetch 2008) narrative is structured into story events. These have core attributes quoted here for clarity:

- *Idea* A brief description of the action that occurs.
- *Actors* The people/creatures/forces that either perform some action or are acted upon.
- *Time* The time at which the action begins.
- *Place* The environment(s) in which the action occurs.
- *Actions* The changes that actors make to themselves, other actors, or their environment.
- *Reasons* The notions held by actors that prompt their actions.

Orthogonally, the choice of these attributes can be classified according to the following design decision properties (Thue, Bulitko, and Spetch 2008):

- *Chooser* The party who made the decision - either the player or the author.
- *Time* The time at which the authoring decision was made.
- *Method* The mechanism used to make the decision - this may be author imagination, a particular (computer) algorithm, or an in-game player action.
- *Justification* The author's or player's reason for using the method that they chose.

The list of the possible values we will consider for *Chooser* are: player, author and emergent (resulting from an emergent AI driven system). Notice that in (Thue, Bulitko, and Spetch 2008) the emergent value for *Chooser* was not explicitly identified. In (Thue, Bulitko, and Spetch 2008) decision time can either be offline (before system deployment) or online (after system is deployed authorship still affects story decisions). For the considered systems it is always offline in case of an author choice, and online for both player and emergent decisions. We will describe the attributes according to what is directly supported by each framework (hypothetically all features are available since both systems interface with Java code). The comparative analysis is summarized in Figure 2.

Regarding the *Idea*, *Actors* and *Place* for both frameworks we have that the chooser is the author, and both method and justification have no restrictions or are biased in any direction. FAtiMA has been applied in serious game applications which would lead it for more pedagogically oriented justifications. However, the framework itself does not condition authoring in that direction.

FAtiMA takes into account the player and an emergent system when deciding the events' *Time* property. Events/actions can happen when emotions trigger motivating goals to be activated or when the user takes action. In ABL the system can react to player input each time it evaluates its behavior tree (player) and sequenced behavior can be encoded (author). Concerning the action property, ABL behaviors are selected according to preconditions and specificity defined by the author, and the player can affect WMEs used in action preconditions. In FAtiMA the concatenation of unitary actions results in an emergent action plan to achieve the most relevant goals, and the player's actions affect the emotional state of characters that will in turn affect their actions. Finally, the *Reasons* in ABL are authorially encoded by the link between behaviors and actions, while in FAtiMA the concrete motivation for a sequence of actions is determined dynamically at runtime. Additionally, in FAtiMA one can define personality traits for characters.

By identifying these design decision features we can have a clearer idea of what steps require more effort or have more restrictions. We analyzed the two approaches according to the following modeling dimensions:

- **mapping theory**: if the framework allows existing mapping community concepts directly (direct) or forces adaptation to the framework's concepts (indirect).
- **emotion**: if the framework supports the expression of characters' emotional state.
- **model checking**: how the framework inform about possible problems in the authored input. This dimension is pointed out as an area of potential innovation in (Mehm et al. 2012).
- **variability**: how the framework supports variation on the experiences generated (Chen et al. 2009).
- **policy change**: effort to fine tune generated experiences in a specific direction (inspired in (Chen et al. 2009)).

Given our modeling, we draw some values for the mentioned dimensions (summarized in Figure 2). We found it harder to directly map theoric conflict concepts of organizational management, such as resolution strategies, in FAtiMA. Neither goals or unitary actions in FAtiMA are a good fit for the notion of strategy. On one hand strategies should define acted behavior like unitary actions, on the other they should also encode *Reasons* dependent on character traits, that in FAtiMA can only be specified at the goal level. We were forced to use boolean character attributes to establish this link, which are little more than flags. In ABL we could map a resolution strategy (e.g. dominating) in a type of conflict (e.g. owes conflict) to a behavior fulfilling a goal of resolving that type of conflict (e.g. resolveOwesConflict).

FAtiMA's focus on emotions does not match well the conflict theory we considered. Nevertheless, conflict situations tend to generate emotional responses and FAtiMA generated an emotional state for the characters with little additional effort (e.g. fear emotions caused by threatened goals). Con-

Properties	ABL		FAtiMA	
time (event)	author & player	sequencing behavior tree re-evaluation	emergent & player	emotion triggering
actions (event)	author & player	preconditions and specificity WMEs changed	emergent & player	plans actions affect emotional state
reasons (event)	author	behavior to goal link	emergent & author	plans and emotions character traits
mapping theory	direct		indirect	through emotion concepts
emotion	none		emergent	plan analysis
model checking	early	compilation time	later	runtime XML verification
variability	embedded	randomness between equivalent behaviors	uncontrollable	numeric uncertainty of goal importance comparison
policy change	medium effort	preconditions numeric tuning	harder	numeric tuning of goal importance and other personality attributes

Figure 2: Authoring differences between ABL and FAtiMA

cerning model checking, the fact that XML authoring errors in FAtiMA are only detected at runtime makes debugging slower, namely for *Actions* and *Reasons*. ABL's compiler error checking allowed a faster iterative process.

In regards to variability of *Actions*, when in ABL two behaviors fulfill a goal, have their preconditions met, and have the same specificity, the system selects one random behavior (e.g. choice between dominating or integrating strategies if the character has high concern for self). Thus, variability is embedded on how the behavior choice is made. In opposition, by default FAtiMA ranks plans according to the extent by which they achieve goals, selecting the optimal one. Consequently, at any time only one behavior can be selected, even if two should be equally likely. For instance, if goal A's importance is only slightly more important than B, there should be a close to 50% chance of B being selected, but currently A would have a 100% chance of being selected. There is still variability due to the dynamic influence of the emotional state on the characters decision making, and consequent numeric uncertainty, but it is harder to get an insight on how that variability will occur.

In a related topic, it is harder to fine tune the experience from a design point of view in a specific direction in FAtiMA because so much of the action choice is left to the emotionally driven planner (policy change). It is unclear how each specific numeric importance value, in the goals for instance, will affect the resulting actions (e.g. effect on behavior choice of changing the exact importance on goals). Unlike ABL in which by linking behaviors goals explicitly this fine tuning is more flexible. Nevertheless, there is still some numeric obscureness when it comes to scalar values used in ABL's preconditions, since for an author to understand which behavior will be selected in a certain context she needs to go through all behavior preconditions fulfilling that goal.

Conclusions

In this paper we have described how to model interpersonal conflict in ABL and FAtiMA, and compared design characteristics of both. Generalizing the method used, we conclude that a possible approach to compare the social expressiveness of AI frameworks is to: model example scenarios; specify design decisions of Chooser, Time, Method and Justification for event properties; consider how these design choices support or hinder the modeling dimensions (mapping theory, emotion, model checking, variability and policy change).

In the specific context of interpersonal conflict, we found it was easier to map conflict concepts in ABL and the model checking process was faster. FAtiMA had better support for emotion and other emergent attributes.

From our study we have realized that in both approaches the process of selecting behaviors based on several related, but separately defined, preconditions is abstruse, consequently affecting policy change. To address this issue, we propose structuring preconditions in decision trees for both frameworks. Behaviors that fulfilled the same goal would be the leafs of a decision tree that would take into account personality traits and contextual factors. Besides making the behavior selection more clear, presenting the decision trees visually to designers would help them understand the consequences of their choices. It should of course be noted that in FAtiMA we would not directly have common goals, but rather common post conditions. Alternatively, the decision tree might simply provide an additional scalar bonus to the behavior selection, effectively relaxing a precondition to a scalar factor.

Acknowledgments

This work has been supported, in part, by the FP7 ICT project SIREN (project no: 258453). We would like to thank John Grey for his invaluable introduction to ABL and Reid Swanson for his feedback on modeling conflict.

References

Campos, J.; Martinho, C.; and Paiva, A. 2013. Conflict inside out: A theoretical approach to conflict from an agent point of view. In *in Proceedings of the Twelfth International Conference on Autonomous Agents and Multiagent Systems*.

Chen, S.; Nelson, M.; Sullivan, A.; and Mateas, M. 2009. Evaluating the authorial leverage of drama management. In

Proceedings of the AAAI Spring Symposium on Intelligent Narrative Technologies II.

Damasio, A. 1994. Descartes' error: Emotion, reason, and the human brain.

Dias, J.; Mascarenhas, S.; and Paiva, A. 2011. Fatima modular towards an agent architecture with a generic appraisal framework. In *Proceedings of the International Workshop on Standards for Emotion Modeling*.

Fikes, R. E., and Nilsson, N. J. 1972. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3):189–208.

2012. Immerse. http://games.soe.ucsc.edu/immerse-project-job-opening.

Mateas, M., and Stern, A. 2004. A behavior language: Joint action and behavioral idioms. *Life-like Characters. Tools, Affective Functions and Applications* 194:1–28.

Mateas, M. 1999. An oz-centric review of interactive drama and believable agents. *Artificial intelligence today* 297–328.

McCoy, J.; Treanor, M.; Samuel, B.; Tearse, B.; Mateas, M.; and Wardrip-Fruin, N. 2010. Comme il faut 2: a fully realized model for socially-oriented gameplay. In *Proceedings of the Intelligent Narrative Technologies III Workshop*, 10.

Mehm, F.; Reuter, C.; Göbel, S.; and Steinmetz, R. 2012. Future trends in game authoring tools. In *Entertainment Computing-ICEC 2012*. Springer. 536–541.

Ortony, A.; Clore, G. L.; and Collins, A. 1990. *The cognitive structure of emotions*. Cambridge university press.

Rahim, M. 2010. *Managing conflict in organizations*. Transaction Pub.

Ryan, M.-L. 2006. *Avatars of story*, volume 17.

Swanson, R., and Jhala, A. 2012. Rich computational model of conflict for virtual characters. In *Intelligent Virtual Agents*, 502–504. Springer.

Thue, D.; Bulitko, V.; and Spetch, M. 2008. Making stories player-specific: Delayed authoring in interactive storytelling. In *Interactive Storytelling*. Springer. 230–241.

Wardrip-Fruin, N. 2012. *Expressive Processing: Digital Fictions, Computer Games, and Software Studies*. Reading, Massachusetts: The MIT Press.

Yannakakis, G. N. 2012. Game ai revisited. In *Proceedings of the 9th conference on Computing Frontiers*, 285–292. ACM.