2009

Comprehensive Believable Non Player Characters Creation and Management Tools for Emergent Gameplay

Jiaming You

Comprehensive Believable Non Player Characters Creation and Management Tools for Emergent Gameplay

(Spine title: Believable Non Player Characters Creation and Management)

(Thesis format: Monograph)

by

Jiaming You

Graduate Program

in

Computer Science

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science

The School of Graduate and Postdoctoral Studies

The University of Western Ontario

London, Ontario, Canada

Abstract

This thesis seeks a way to integrate popular psychosocial components required for believability to build a believable Non Player Characters (NPCs) model using the techniques of emergence. The believable NPCs model is scalable in terms of psychosocial models, customizable, flexible and data-driven. Comprehensive believable NPCs creation and management tools were developed to compose, generate, and maintain the system configuration data, as well as NPC profile data, using XML. Furthermore, a run-time prototype has been developed based on our proposed model to test its effectiveness. The prototype has also been evaluated for believable emergent behaviours in different social scenarios.

Keywords: believable NPCs, psychosocial components, emergence, emergent behaviour, customization, data-driven, tools.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Michael Katchabaw, for his inspirational guidance, timely feedback, useful resources and the passionate encouragement he gave me throughout the development of this work. Without his endless support and concern, this thesis would not have been possible.

I am also very grateful to Dr. James Andrews for his early feedback on the proposal of this work. His invaluable suggestions made me realize the overlooks and finally helped to strengthen this thesis statement.

Many thanks to my colleagues and friends for their constructive ideas and constant concern.

Last but not least, a special thank you to my parents for their continuous care, support and for accompanying me on this journey.

Dedication

to my parents

Contents

List of Figures

Chapter 1

1. Introduction

With the rapid development of the game industry, more realistic Artificial Intelligence (AI), more dynamic gameplay, and more interactive game world are needed for games. Much of a game's AI is responsible for Non Player Characters (NPCs) behaviour control. NPCs populate the game world and fill any role not occupied by the player. Therefore, they can play an important part in a considerable number of games. The traditional ways to control NPCs' behaviours are scripting and hard-coding in order to reduce computation complexity. Such scripted or hard-coded behaviour tends to be so predictable, however, that it easily breaks players' suspension of disbelief. To achieve believability, an NPC model must be developed to be dynamic with psychosocial components (Loyall, 1997).

This thesis addresses the above problems, by providing a believable NPC model with integrated psychosocial components using the techniques of emergence, as well as comprehensive believable NPCs creation and management tools to allow for easy development and customization of NPCs. Our believable NPCs model would be mainly used in open world games (*Grand Theft Auto*, etc), role-playing games (RPGs), action/adventure games and games heavily dependent upon their stories, characters and so on.

1.1. Problem Statement

There have been many architectures developed focusing on different aspects of psychosocial components. However, few, if any, of them have taken all of the psychosocial components required for believability into account. Even though some generic agent architectures have simple unified psychosocial behaviour in NPCs, how to integrate different popular psychosocial component models into the architecture

and how to apply these sophisticated models to facilitate the rapid game development are not mentioned. This, unfortunately, leaves a very large and difficult job for developers; one that is quite likely outside of their traditional areas of expertise.

Consequently, it is proposed that this thesis will investigate how to integrate all the necessary prevalent psychosocial component models into a believable NPC architecture to facilitate emergent gameplay, and then design and implement comprehensive believable NPCs creation and management tools to help game developers avoid this non-trivial work.

1.2. Background

Before delving into our proposed solution, it is helpful to briefly overview the fundamentals of emergence and social psychology, as these elements play a key role in our solution. In this section, we will provide an introduction to the concept of emergence and key psychosocial models. This background information will provide a basic idea about the techniques and theories used in our research. (We will explore this background and other related work in this area in further detail in the next chapter of this thesis.)

1.2.1. Emergence

Emergence is a concept that the properties, behaviours, and structure that occur at higher levels of a system, are not present or predictable at lower levels (Sweetser, 2008). In other words, by defining many simple, independent rules on how lower level objects interact with each other, global dynamic phenomena which are not specifically programmed into the system emerge from those local interactions. Emergence allows the system to have more flexible and dynamic responses instead of the static and predictable reactions due to the scripting or hard-coding. Smith and Smith gave a real world emergence example on how global warming results in cooler

weather in England (Smith and Smith, 2004).

Global warming → Polar ice caps melt → Ocean Currents Shift → Less warm air arrives in England → England gets colder

Pfeifer defines emergent gameplay as "a large amount of gameplay experiences from a much smaller set of interconnected game rules" (Pfeifer, 2004). Board games, for example, although the rules within the game are quite simple, themselves are complex. For this reason, many board games and card games have remained popular for centuries, with endless replayability. From development efficiency and economic perspectives, scripting every possible situation exhaustively in game is very difficult and time consuming (Wootton, 2006). On the other hand, all emergence needs is the initial effort to set up rules, after which developers can benefit from its ability to support a variety of undefined circumstances that the player may create within the game.

1.2.2. Social Psychology

Loyall (Loyall, 1997) did research on believable agents and pointed out that the requirements for believability are the following psychosocial components: personality, emotion, self motivation, change, social relationships, consistency and illusion of life (appearance of goals, concurrent pursuit of goals, parallel action, being reactive and responsive, being situated, having a resource bounded-body and mind, existing in a social context, being broadly capable, and being well integrated (in terms of capabilities and behaviours)). The concept and requirements of believability from Loyall will be used to build our believable NPC model.

People have long been trying to seek the logic behind passion, and therefore much of research has been done on psychosocial concept modeling, including personality models, emotion models, motivation models and so on. Some of the more important and relevant of those models will be expanded on later in the next chapter.

One of the famous personality models for testing job tendency is the Myers-Briggs Typology, which is built on four pairs of traits that are considered complementary and distinct (Caplinger, 2000). The Myers-Briggs system measures people in the areas of: how a person relates to others (either by Extraversion or Introversion), how a person takes in information (either by Sensing or iNtuition), how a person makes decisions (either by Thinking or Feeling), and how a person orders their life (either by Judging or Perceiving). The Five Factor Model (FFM), also known as OCEAN, is another popular personality model, evaluating personality in five independent dimensions: Openness, Conscientiousness, Extraversion, Agreeableness and Neuroticism (McCrae and John, 1992). The Myers-Briggs Typology and FFM each have a different focus, yet they are not totally separated, since there is some overlap on personality traits between them. EI and SN are strongly correlated to extraversion and openness respectively, while TF and JP are moderately related to agreeableness and conscientiousness respectively (McCrae and Costa, 1989). In the theory from Reiss, a character's personality can be described as a set of weighted degrees of the following desires: power, curiosity, independence, status, social contact, vengeance, honor, idealism, physical exercise, romance, family, order, eating, acceptance, tranquility and saving (Reiss, 2004).

Likewise, there are many different emotion models proposed by different theorists. Among those models, one of the basic emotion models is from Plutchik. In his emotion theory, basic emotion pairs are opposite to each other: joy and sadness, acceptance and disgust, fear and anger, surprise and anticipation. Furthermore, he also identified eight advanced emotions, and each one is composed of two basic ones (Plutchik, 2001). Parrott proposed a categorized, tree structured list of emotions, which consists of primary emotion, secondary emotion and tertiary emotion in a hierarchical fashion (Parrott, 2001). A model mainly for universal facial expressions is from Ekman, Friesen, and Ellsworth, which includes anger, disgust, fear, joy, sadness, and surprise (Ekman et al., 1972).

As for motivation, Maslow developed a well-known hierarchy of needs, including survival need, security need, social need, needs of psychological safety, cognitive need, aesthetic need, self-fulfillment and self-actualization (Maslow, 1946).

Other interesting psychosocial concepts are coping and stress. Coping involves two aspects: solving the problem (problem focus) and addressing the emotions (emotion focus) (Beaumont, 2005-2008a). Coping can be viewed as reducing the suffering stress in some sense, which is caused by stressors. Stress is something intrinsic that connects all human behaviours together throughout their lives. All problems faced and events occurring may cause more or less stress, either physical stress or emotional stress, depending on context. Coping is what we do to solve the problems and to address the issues at hand, and finally let ourselves stay calm and have peace once again.

Finally, the life guidance system presents an overview of how those psychosocial components can be integrated together to guide us through our daily lives (Beaumont, 2005-2008b).

1.3. Proposed Solution

In this section, we outline the techniques, psychosocial models and methodology used in our research to solve the proposed problem of achieving believable psychosocial behaviour in NPCs.

Bailey's research work laid the foundations and provided a core framework for believable NPCs (Bailey, 2007). Unfortunately, this work did not extensively explore the details of psychosocial models available for use. Hence, it is left to others to determine how to integrate the popular psychosocial component models into this framework and make it useable to game developers. However, this is a considerable

amount of work, requiring domain expertise that is outside the traditional background of most game developers. Thus, constructing comprehensive NPC models and developing robust and flexible believable NPCs creation and management tools to facilitate game developers' efforts are clearly necessary.

As discussed before, scripting and hard-coded state machines require game designers to know and anticipate every situation that may occur within their games, which is excessively time-consuming and very complex to exhaustively define. Therefore, emergence will be used to develop a run-time system for believable NPC behaviours, as was done in the framework originally created by Bailey.

As for psychosocial models, the Myers-Briggs Typology (Bateman and Boon, 2006) is very suitable for modeling personality in games, because it is easy to build the linkage between occupation tendency and character personality; secondly the personality traits coincide with human cognitive processes and a sensing/thinking/acting architecture. In the Myers-Briggs Typology, EI can affect how NPCs behave in social networks, SN can affect how NPCs feel about others' behaviours, TF can influence NPCs' reasoning processes, and JP can change an NPC's goal organization. This means that when an NPC is executing sensing, one can focus on SN traits, thinking can focus on TF traits, goal organization can focus on JP traits, and social interaction can focus on EI traits. This has the potential to greatly reduce the complexity of computing NPC behaviours. Since it is quite possible that the eight Myers-Briggs personality traits are too abstract, we can further subdivide the eight personality traits into more detailed descriptions, and it implement varied levels of detail in creating believable NPCs.

However, the Myers-Briggs Typology alone is not diverse enough for modeling all kinds of characters with different characteristics, since the Myers-Briggs Typology is mainly focused on human cognitive processes and linkage with profession choice. If we were to create someone who is rude, selfish and jealous, which are independent

from occupation, we would need to turn to another personality model for help, for example FFM. However, some adjustment may be needed since there are some overlaps on personality traits between them. Due to the fact that there is no single model that comprehensively covers every personality trait, we take an integrated, multi-model approach that brings together elements from many models at once, and we provide the flexibility for users to customize them as necessary. Furthermore, it is interesting to note that the five factors in FFM were derived as personality indicators from Cattell's larger 16 personality factors set (16PF), which means FFM can be represented using a more detailed low-level personality model as needed (Syque, 2002-2008; Sinclair). The correlations that FFM has with other personality models give us the potential to scale our models and system in both horizontal and vertical ways.

From an emotion perspective, using Plutchik's basic emotions and advanced emotions may be a good solution to modeling emotion for games. Offering different levels of detail to users can also be achieved by placing the appropriate emotions from Parrott's tree structured list into a state transition diagram to emulate subtle emotional changes. In the same fashion as modeling personality, an integrated, multi-model, scalable approach which allows customization is used.

Maslow's hierarchy of needs, as well as stress and coping mechanisms, are also supported by our models. Last but not least, the integration of all of the psychosocial component models will use the life guidance system as a point of reference (Beaumont, 2005-2008b).

1.4. Outline

Chapter 2 will present a review of previous work in this area, beginning with an overview of the reasons and techniques for building believable NPCs in games, the use of emergence in game, agent architectures and the research done on believable

NPC modeling, and ending with a discussion of the reviewed architectures and models. Chapter 3 will present the design and conceptual modeling of this work (the design overview, the NPC data model design, NPCs tools design, and emergent run-time system design). Chapter 4 outlines a prototype that was implemented to evaluate the design, and Chapter 5 will discuss this evaluation. Finally Chapter 6 concludes with final remarks, a review of contributions made by our work, and areas for future work in this area.

Chapter 2

2. Literature Review

The focus of this chapter is to review what research has previously been done in the area of believable NPC modeling, and identify open problems and issues still needing to be addressed.

Section 2.1 gives an overview of why we need believable NPCs and how to create believable NPCs. Section 2.2 discusses emergence in games, with a focus on the concept of emergence and the techniques to create emergence, and how to generate emergent gameplay in video games. Section 2.3 starts by discussing generic agent architectures and hybrid agent architectures, and then discusses the core concepts of believability in NPCs, and the efforts made towards the creation of believable NPCs and agents. The review concludes in Section 2.4 with a discussion of the advantages and deficiencies of the research work reviewed and the research opportunities left, as well as the contributions and impact that will be made by our current work.

2.1. Overview

Why do we need believable NPCs? As discussed before, the traditional ways to control NPCs AI, scripting and hard-coding, are too predictable and limiting to immerse players of modern video games.

How do we create believable NPCs? First of all, we need emergence, which is an approach allowing high-level flexible behaviours to emerge dynamically from low-level building blocks. In the end, traditional approaches to NPCs, as discussed above, lack the ability to create realistic psychosocial behaviours, while emergence has the potential to do so much more effectively. Secondly, we need psychosocial

components (personality, emotion, self motivation, change, social relationships, consistency and illusion of life) as our low-level building blocks in the emergent system (Loyall, 1997). Figure 2.1 describes the approach we took to build believable NPCs.

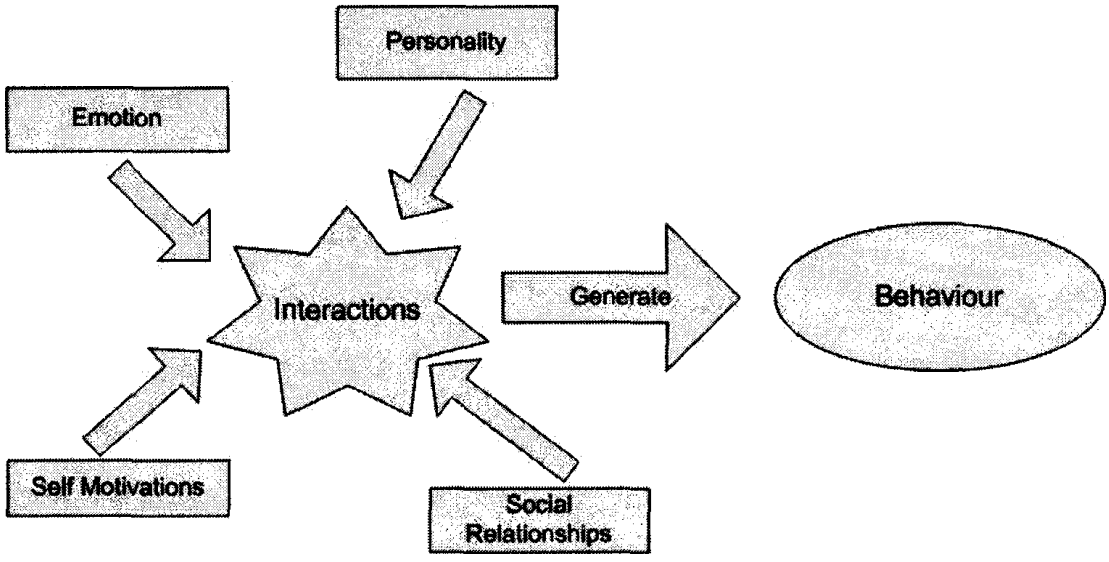

Figure 2.1: The way we used to create believable NPCs.

2.2. Emergence

Emergence in games is a topic that has attracted attention in recent years. Developers think that by introducing emergence into gameplay, we are more likely to create gameplay that is more natural and flexible, where players can act on their own only with limitation of their imagination and creativity (Sweetser, 2008).

2.2.1. Emergent Gameplay

From Sweetser's point of view (Sweetser, 2008), emergent gameplay can be achieved by defining simple, local rules, behaviours, and properties for game objects and their interactions with the game world. Emergent gameplay occurs when interactions between objects in the game world result in a second order of consequence that was not planned, or perhaps predicted by the game developers, yet the game behaves in a rational way that makes sense.

In fact, several of the most successful and enjoyable games have elements of emergence; for example, the rules in board games, the strategies in *SimCity*, the environments in *The Sims*, the mutations in *Spore,* and so on.

Wootton believe that by implementing emergence in game design, we can benefit from increased realism, better longevity and gaming experience, higher development efficiency and economics (Wootton, 2006).

Emergent gameplay makes the game world more interactive, reactive, intuitive and realistic, bringing in many possibilities and flexibility into the game world, which greatly enhances the immersion for the players.

2.2.2. Emergence and Game Design

Wootton has discussed many current issues in game design in (Wootton, 2006). Although much inspiration has been taken from biology (e.g., the use of neural networks and genetic algorithms), developers still tend to implement high-level behaviours such as cooperation and interaction in a hard-coded and top-down fashion. The problem is that coding or scripting these behaviours will require tremendous effort in handling complexity, robustness and completeness issues as players demand a more and more realistic gaming experience. Emergence, a bottom-up approach, can give the developer a better way to create high-level human-like behaviours and interactions if used properly.

2.2.3. Why Emergence in Games

Why do we need emergence in games? There are two main reasons for putting emergence in games. One reason is that emergence allows for a more flexible and dynamic game world to engage the players. Another reason is that emergence has

higher development efficiency and it is more economical, without the need to consider every possible situation exhaustively for a game, which is very difficult and time consuming (Wootton, 2006).

2.2.4. Techniques for Emergence

Sweetser talked about two contrasting approaches in designing a game world in (Sweetser et al., 2005). One is scripting, which requires programmers to anticipate, hand-craft and script specific game objects, events and player interactions. The other one is emergence, involving defining general, global rules that interact to produce emergent gameplay.

Different techniques can be used for different implementations of games, defining whether the system is static and scripted or dynamic and emergent (Sweetser et al., 2005). Techniques that require everything to be built into the system during development, with no room for adaptation or unexpected behaviour, can only give rise to a system that behaves exactly as how it was told to behave. On the other hand, techniques that are only given the boundaries for behaviour by developers, instead of a fixed script, or are otherwise able to grow and change, can facilitate interesting and unexpected behaviours (Sweetser et al., 2005).

Emergent behaviour occurs when simple, independent rules interact to give rise to behaviour that was not specifically programmed into the system (Rabin, 2004). Techniques that can be used to facilitate emergence come from complex systems, machine learning and artificial life (Sweetser et al., 2005).

2.2.4.1. Fuzzy Logic

Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with reasoning that is approximate rather than precise (Klir et al., 1997). Fuzzy logic

allows greater flexibility, variation, and non-determinism than simple linear techniques, because it can have intermediate values defined between conventional values, such as yes/no or true/false (Sweetser, 2008).

Fuzzy logic is explained at length in (Sweetser, 2008). It can be very useful in decision-making, behaviour selection and NPC emotional state emulation in game systems. For instance, fuzzy logic can be used for NPCs to decide how they feel about other characters, or to decide how emotional they are. With the ability to represent a concept using a small number of fuzzy values, fuzzy logic can be powerful. On the other hand, in Boolean logic, every state and transition has to be hard-coded. Fuzzy logic is good at solving problems that are non-linear, where no simple mathematical model is available. However, it is not the best solution if there is a simple solution and it is complicated to build the model from scratch.

2.2.4.2. Cellular Automata

Sweetser discussed the potential applications of Cellular Automata (CA) in games in (Sweetser et al., 2005). CA is a widely-used technique in the area of complex systems to study agents and their interactions. A traditional CA is a spatial, discrete time model in which space is represented as a uniform grid (Bar-Yam, 2003). Each cell has a state, which will be updated according to a set of rules at every discrete time step. The state of a cell at time t is a function of the states of its neighboring cells and itself at time t−1. These neighboring cells are a selection of cells relative to the specified cell. For example, in one-dimensional CA, each cell typically has two neighboring cells, those are cells to its left and right. In the similar manner, a two-dimensional CA can have four or eight neighbors depending on whether considering cells diagonally adjacent to a cell as neighbors. Many current computer games have a static environment which can be improved by using CA, because the use of CA could give rise to more dynamic and reactive environmental game objects that are currently scripted, such as fire, water, explosions, smoke, and heat.

Influence mapping, a variation of CA, is a way to represent the distribution of power within a game world in a two-dimensional grid (Rabin, 2004). Designers can use influence maps for strategic assessment and decision-making in games (Sweetser, 2001). In the game *SimCity*, designers used it to model the influence of various social entities, such as police and fire stations around the city (Rabin, 2004).

2.2.5. Emergence in Games

In board games, the players are given a finite space (the board), a set of objects (the pieces), and a set of simple rules (Sweetser, 2008). The simple elements within the games give rise to almost unlimited possibilities and emergent gameplay. Similar to board games, card games have a simple set of elements, along with a simple set of rules of play, producing unpredictable and emergent gameplay.

In the following sections, we present the potential areas where emergent gameplay can be produced and how it can be accomplished and used effectively. Sweetser has examined these issues at length in (Sweetser, 2008).

2.2.5.1. Game Worlds

Game worlds are the possibility space of games, including the space, terrain, objects, physics, and possible interactions that can be carried out inside the games (Sweetser, 2008). In order to create an emergent game world, one must design different types of rules of local interactions between objects, rather than specific, scripted and rigid global rules. Defining a set of simple, general elements and rules that can give rise to a wide variety of interesting, challenging behaviours and interactions in different situations is the key to creating emergent game worlds. Typically, this involves the use of a physics system to manage collisions between objects, the exertion of forces in the world (such as gravity), and so on.

2.2.5.2. Characters and Agents

Characters and agents are important types of objects in game worlds as they give the game life, story, and atmosphere (Sweetser, 2008). Without them, the game world would be less rich, less interesting and plain. This is especially true for RPGs; this type of game includes varied characters that fill a wide range of different roles in society, from kings to vagrants. More than anything else in the game world, players expect lifelike behaviours from their game characters.

Players expect game characters and agents to behave believably and intelligently by being cunning, flexible, unpredictable, adaptable and realistic (Sweetser et al., 2003). However, players often find that characters in games are not believable and predictable (Sweetser, 2008). Characters' actions and reactions in games should be able to demonstrate their internal states and awareness of events in their immediate surroundings (Drennan et al., 2004). However, characters in most of the current games are scripted to behave exactly the same regardless of their internal states or the surrounding environment. The more responsive, reactive, adaptive and dynamic the characters and agents in games, the more lifelike, believable, and diverse the game worlds will be when presented to the players (Sweetser, 2008).

Characters play a vital role in creating an emergent game world. Introducing entities that have a decision-making process of choosing how to react to the changing internal states and environment amplifies the variation and unpredictability of a system (Sweetser, 2008). By adding a new level of complexity and variation to the game world, reactive agents can largely extend emergent behaviours and thus emergent gameplay. As agents are able to choose how to react to the environment, they are able to actively change the state of the world in ways that might not have occurred without their intervention. Furthermore, differences between individuals, such as goals,

motivations, personality, emotions, values, belief, social status, social roles, relationship, reputation, attitudes and so on, can add further variation and complexity, resulting into a more dynamic and emergent game world. Not only can agents choose how to react to a given situation, different agents will choose to react in different ways in the same situation (Sweetser, 2008).

Wootton also pointed out that when designing for emergence, goal-directed individuals are usually desirable for the systems (Wootton, 2006).

Characters and agents can be used to create emergence in games by being given an awareness of their environment and internal states, and an ability to react to the changing environment according to their internal states (Sweetser, 2008). In this case, the agents then can sense (information taking), react to (decision-making), and alter (self adaptation). By creating models for internal agent states and the external environment of the game world, and setting up a set of rules for reactions between elements of these models, characters and agents in the game can follow simple rules of behaviour, taking into account the complex environment around them. In doing so, they become emergent characters in the game world, making a more believable and immersive game world overall.

2.2.5.3. Emergent Narrative

A game's narrative is the story that is being told, unfolded, or created as the player makes his/her own way through the game (Sweetser, 2008). Instead of being scripted or fixed in advance, this narrative could be the product of the interactions carried out by the players in the game world. In doing so, the story is created by the players about their characters and the challenges they faced as they played the game. To create emergent narrative, the developer is ultimately tailoring the narrative to put the player in the center position in the story and allowing the player to drive the story, instead of

forcing the story to drive the player.

2.2.5.4. Social Emergence

Among all the possibilities for emergence in games, social emergence is the most complex and unpredictable in current games (Sweetser, 2008). When several millions of people get together to play the massively multiplayer online role-playing games (MMORPGs), such as *World of Warcraft*, the result is as complex and divergent as a large city. Instead of trying to find out how to create emergent social systems in games, it is more suitable to understand, model, construct and support the complexities that come out naturally from human interactions by looking into psychology, sociology, economics, and even law.

From what we discussed in this section, the connection between emergence in games and believable NPCs is that emergent gameplay need believable NPCs involved, and believable NPCs need emergent techniques to be developed. Consequently, we will review the research work done on believable NPC architectures in next section.

2.3. Believable NPC Modeling

There are four key areas of games that hold the potential for emergent gameplay: the game world, characters and agents, narrative, and social systems as described before.

NPCs are a significant part of most games, in which they fill a wide range of roles, such as shopkeepers and barkeepers, populating the fantasy game world to make it more realistic to the players. NPCs are also necessary for massively multiplayer online games (MMOGs), as there are many non-hero roles that human players do not want to fill in. However, with predictable behaviours and completely scripted dialogue, most of the NPCs have no awareness of occurring events and psychosocial

states. NPCs rarely show emotion and consistent personality in the game world, not to mention the illusion of life, which can break the suspension of disbelief of the players.

This section will focus on how to build a believable NPC model for emergent gameplay and the research that has been done in this area.

2.3.1. Agent Architectures

Travis claimed that an NPC architecture should be highly reusable, sophisticated in AI for action selection, able to turn off the features that are not required, and able to emphasize the human limitations and temporal constraints (Freed et al., 2000). In addition to these traits, Baillie-De Byl proposed that such an architecture should also consider the repeatedly executed tasks of perception, inference, action selection and behaviour (Baillie-De Byl, 2004).

2.3.1.1. Generic Agent Architectures

Baillie-De Byl identified five generic agent architectures that can be applied to NPCs creation in games: reactive, triggering, goal-based, utility-based, and anytime architectures (Baillie-De Byl, 2004).

The reactive agent architecture, also known as a reflex, or stimulus-response architecture, is the simplest agent architecture (Baillie-De Byl, 2004). The reactive agent architecture does not model the entire mind of a human; instead, it only tries to represent the quick thinking instinctual behaviours. The advantages of reactive agent architecture are speed; agents can react very quickly. However, the drawbacks are that the developers must know every possible environmental state and game event to create rules to allow the agents to respond to them. These kinds of agent architectures are implemented by hard-coding rules without holding state, and thus have no long-term reasoning (Bailey, 2007).

A simple example behaviour for a reactive agent would be as follows. Assuming there is an agent guarding an area of the environment, one of its knowledge base rules is (Baillie-De Byl, 2004):
Rule: if hear (noise) then action (activate_alarm)

The triggering agent architecture is basically the same as the reactive agent architecture; the only difference is that the triggering agent architecture allows agents to maintain internal states (Baillie-De Byl, 2004). In these types of architectures, agents can have memory and some form of self-awareness (Bailey, 2007).

Similarly, a rule guiding the guard would be like this in triggering agent architecture (Baillie-De Byl, 2004):
Rule: if hear (noise) and state (awake) then action (activate_alarm)

Both reactive and triggering agent architectures are fast, but both of them need to be scripted for all possible situations and neither of them can be easily scaled up (Bailey, 2007).

The goal-based agent architecture holds a set of preferred states which define the agent's goals and desires (Baillie-De Byl, 2004). These kinds of agents need extra reasoning steps in the inference module as the agents must decide how a perceived stimulus affects their own goals and associated actions and then plan to achieve their goals. By allowing the agents to plan for their own goals, agents can make flexible plans, and it is quite easy to add more plans for a specific goal into the knowledge base without much modification work. However, the disadvantage is that the goals are satisfied on a first-come, first-served basis, thus the agents are not able to identify which plans are better than the others in fulfilling the same goal.

The rules guiding the guard would be like this in goal-based agent architecture

(Baillie-De Byl, 2004):

Rule 1: if hear (noise) then goal (alert_others)

Rule 2: if goal (alert_others) then goal (activate_alarm)

Rule 3: if goal (alert_others) then goal (use_radio)

Rule 4: if goal (activate_alarm) then plan (locate_alarm, move_to_alarm, sound_alarm)

Rule 5: if goal (use_radio) then plan (pick_up_radio, set_frequency, call_for_help)

Rule 6: if plan (x,y,z) then action(x), action(y), action(z)

Rule 7: if hear (alarm) then ¬goal (activate_alarm)

The utility-based agent architecture improves upon the goal-based agent architecture by placing importance ratings to goals dynamically according to the agent's current situation (Baillie-De Byl, 2004). However, this could be a problem as the goal hierarchy becomes more and more complex, because the computation load could grow to be much heavier than a real-time interactive game could afford.

The anytime agent architecture has a time limit in the inference and action selection module on top of the utility-based agent architecture. This allows processing to be limited or capped, which can yield benefits in cases of not having to compute the entire plan which may at some later time become irrelevant (Baillie-De Byl, 2004).

2.3.1.2. Hybrid Agent Architectures

Baillie-De Byl also discussed hybrid agent architectures for NPCs, including the Soar architecture, the H-CogAff architecture, the EXCALIBUR Project, and the being-in-the-world architecture in (Baillie-De Byl, 2004).

The Soar architecture is based on the utility-based agent architecture, and is able to learn new rules (Baillie-De Byl, 2004). There are five main modules: Chunking

(which is responsible for learning new rules), Long Term Memory (which defines agents' knowledge by a set of rules), Working Memory (which generates agents' goals), Production Matching (which decides which rules are fired), and Preference List (which is used to order the goals and associated actions). To date, the Soar architecture has been used in a death match version of *Quake II*.

The H-CogAff architecture is based on the anytime agent architecture, and is a three layered architecture designed to emulate the model of human information processing (Baillie-De Byl, 2004). The Reactive Processes layer works as a reactive agent that takes in internal and external stimuli and gives out quick and crude response when extra processing time is unavailable. The Deliberative Processes layer works as goal-based agent that reasons about its action for achieving its goals. The Meta Management layer is responsible for self-reflection, self-monitoring of internal states and allowing the agent to evaluate its behaviour to modify its thought processes. A Personae module is also attached in H-CogAff architecture to allow agents with different personalities to behave differently.

The EXCALIBUR Project developed a generic NPC architecture which is applicable in a complex game environment (Nareyek, 2000). The EXCALIBUR architecture is based on the anytime agent architecture and considers temporal scheduling (making sure the actions are still valid when they are carried out), the NPC's need for environmental resources, and the ability to plan with incomplete knowledge of the game world (Baillie-De Byl, 2004).

The Being-in-the-World architecture has two modules: a real-time coping module and a reasoning module (Baillie-De Byl, 2004). The real-time coping module is responsible for perception, managing a knowledge base, fast stimulus-response behaviours, and forming plans into concrete actions. The reasoning module is responsible for using the current game world states to generate a plan of low-level goals to be completed for the real-time coping module. Both modules run in parallel

and use a queue of goals to communicate.

2.3.2. Believable NPC Architectures

According to Loyall (Loyall, 1997), the requirements for believability in an artificial being are personality, emotion, self-motivation, change, social relationship, and the illusion of life (appearance of goals, concurrent pursuit of goals, parallel action, being reactive and responsive, being situated, having a resource bounded-body and mind, existing in a social context, being broadly capable, and being well integrated (in terms of capabilities and behaviours)).

Personality: A rich personality should influence everything that a character does, from talking to thinking. It is the unique ways of doing things make characters interesting. Personality is about the unique and specific, not the general.

Emotion: Characters exhibit their own emotions and response to the emotions of others in personality-specific ways.

Self-motivation: Characters do not just react to the outward stimuli. They have their own internal drivers and desires to pursue whether or not others are interacting with them.

Change: Characters grow and change with time, in a manner consistent with their personality.

Social relationships: Characters engage in detailed interactions with others in a manner consistent with their relationships. In turn, these relationships change as a result of the interactions.

Illusion of life: This is a collection of requirements such as: pursuing multiple,

simultaneous goals and actions, having broad capabilities (e.g. movement, perception, memory, language), and reacting quickly to stimuli in the environment.

Traditional character artists do not mention these requirements explicitly, because they often get them for free (from a human actor, or as a deep assumption in animation). But builders of interactive characters must concern themselves explicitly with building agent architectures that support these requirements.

However, from the agent architectures discussed above, most of them are mainly concerned about cognitive processes, and so they should be considered purely cognitive architectures (Bailey, 2007). Lawson also point out that developers are restricting themselves in using cognitive science to model NPCs in a letter to Gamasutra (Lawson, 2003). The possible exceptions are the utility-based agent architecture and those architectures based on it, like the Soar architecture (Bailey, 2007). The only factor that distinguishes the utility-based agent architecture from others is that the agents make decisions due to internal stimuli. Even the utility-based agent architecture, the Soar architecture, the H-CogAff architecture, the EXCALIBUR architecture and the being-in-the-world architectures could support emotional simulation; however none of them appear to consider social behaviours.

The following sections discuss the work has been done in the areas of emotions modeling, personalities modeling, social relationships modeling and so on.

2.3.2.1. Emotions Modeling

Affective computing, mainly concerned about emotion in computing, is defined as "computing that relates to, arises from, or deliberately influences emotion or other affective phenomena" by Picard (Picard, 1997). In the area of affective computing, much work is being done to develop AI capable of identifying, processing, and

synthesizing emotions (Baillie-De Byl, 2004; Picard, 1997). However, the area of NPC modeling in video games is currently mainly concerned with synthesis of emotional states and usually for video games, only emotional states that will directly affect a character's action selection should be modeled (Funge, 2004; Baillie-De Byl, 2004). Emotion is a fundamental requirement of believability and it is expressed by a character's outward appearance and behaviours and used internally for perception, reasoning, and decision-making (Baillie-De Byl, 2002).

Tomlinson discussed two main models of emotion in computing: a categorical approach and a dimensional approach (Tomlinson and Blumberg, 2002). The categorical approach separates emotional phenomena into a set of basic emotions, whereas the dimensional approach maps a range of emotional phenomena onto an explicitly dimensioned space (Tomlinson and Blumberg, 2002). One example for categorical emotions is the basic emotions Ekman brought forward: anger, disgust, fear, joy, sadness and surprise (Ekman, 2005). An example for the dimensional approach is that Breazeal used a three-dimensional space (Arousal, Valence, Stance) to give affective tags to occurrences perceived by her robot, Kismet (Breazeal, 2000). The techniques that are usually used to model NPC mind states, like finite state machines (FSMs), can also be applied to model categorical emotions, since the differentiation between categorical emotional states is fairly discrete (Picard, 1997). Picard also suggested that different emotional states have different chances of transitions into other emotional states depending on the states involved – for instance, it is more likely to transit from interest to joy than from distress to joy (Picard, 1997).

Baillie-De Byl theorized that emotion could be divided into five different categories according to the use of emotion in NPCs (Baillie-De Byl, 2004). Those five categories are emotional behaviour, fast primary emotions, emotional experience, body-mind interactions, and cognitively generated emotions. The area of emotional behaviour refers to systems that are capable of appearing emotional even they do not have the power to internally process emotional response as humans do. Fast primary emotions

mainly focus on implementing survival mechanisms with which an NPC could quickly react to protect itself, such as dodging attacks. Emotional experience refers to the ability of being aware of its own emotional state and subjective feelings; this is difficult to implement due to its biochemical nature. Emotional experience may have no relevance with games, because it does not always have a direct impact on external behaviours of NPCs (Funge, 2004; Baillie-De Byl, 2004). Body-mind interactions refer to modeling how emotions influence and change body states, and vice versa (Baillie-De Byl, 2004). Canamero implemented this in his research in which NPCs with internal states constructed from synthetic hormones can choose to attack, withdraw, drink, eat, play, rest and each behaviour has an effect on the NPC's internal states (Canamero, 1997; Baillie-De Byl, 2004). Finally cognitively generated emotions can be explained as the emotions generated on appraisal of a stimulus (Baillie-De Byl, 2004). One famous model in classifying cognitively generated emotional states is from Ortony, Clore and Collins (OCC), whose model tests the valence reaction (either a positive or negative response) to one of three elements: the consequences of an event, the actions of the agent, and the aspects of objects (Ortony et al., 1988; Baillie-De Byl, 2004). The Oz Project used this OCC model to implement the Tok agent architecture, which has two subsystems interacting with each other, Em and Hap (Bates et al., 1994). Em is primarily responsible for generating different intensities of emotions according to the perceived importance of a goal (Baillie-De Byl, 2004). Unlike Em, which is for emotions, attitudes, and standards, Hap is responsible for goals and behaviours (Bates et al., 1994).

Paanakker talked about an emotion component, a software component that can be integrated with other AI processes to give characters emotions (Paanakker, 2008). The idea is that humans have coloured perceptions and irrational behaviours under the influence of emotions. An instant fight or flight response can be triggered without much deep logical thinking. The placement of the Emotion Component in the AI architecture is shown in Figure 2.2.

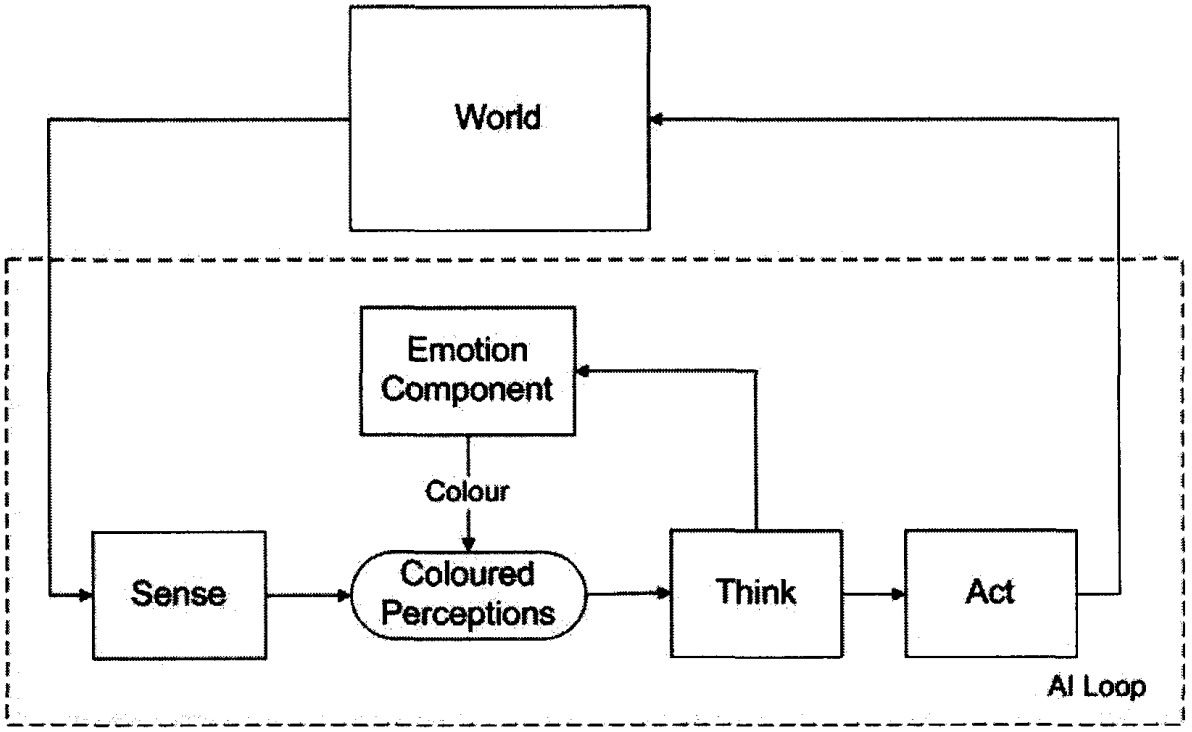

Figure 2.2: The placement of the Emotion Component in the AI architecture (Paanakker, 2008).

Paanakker also discussed at length emotion activation functions (Paanakker, 2008). Different activation functions can be assigned to different emotions based on how well they match each other. For example, as shown in Figure 2.3, E can be used for impatience, C for excitement, D for fear. However, for love, some people might prefer F for a love that grows over time, each time at a small step deepening the feeling, while others might prefer D or B for falling in love at first sight. Making subtle different emotion activation functions for different characters can add another layer of believability and emergence.

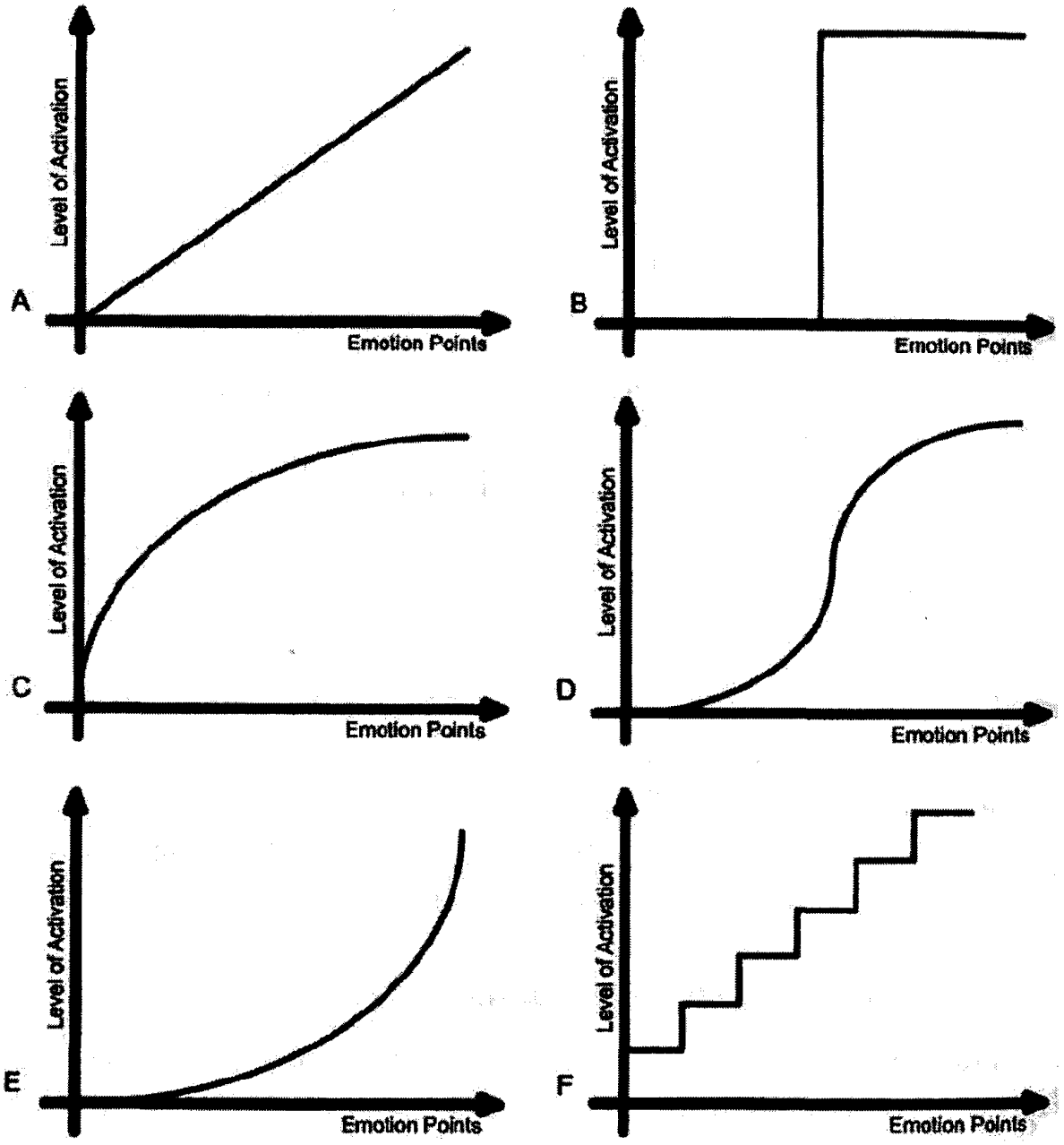

Figure 2.3: Different activation functions (Paanakker, 2008).

In recent years, people have developed several agent architectures capable of processing emotions in various ways.

Zhou proposed an interesting emotion transition model for an NPC based on the PAD emotional state model, which is a three-dimensional emotion space consisting of three nearly independent dimensions to describe and measure emotional states: pleasure/displeasure, arousal/non-arousal, and dominance/submissiveness (Zhou et al.,

2006). The emotion transition model allows three kinds of objects in game: NPC, player, and scene, and each of them is embodied with a vector of PAD (Mehrabian, 1995) valence scores. A game (NPC, player, scene) was defined as one relationship in the model, along with some equations defining how emotion valence scores get updated for the characters inside the model. With distributing different PAD values to different scenes, this model can simulate different emotional reactions of NPC in different surroundings in an easily scalable way.

COGNITIVA is a multi-layered architecture, which provides agents with emotion-influenced behaviour (Imbert and Antonio, 2005a, 2005b; de Freitas et al., 2007). COGNITIVA covers three kinds of behaviours: reactive, deliberative and social, and each of them comes from a related architecture layer. The reactive layer is responsible for immediate responses to changes in its environment. The deliberative layer provides goal-directed behaviour, and the social layer provides goal-directed behaviour considering the existence of other agents. The interaction of these three layers with the other two modules, the sensors and the effectors, is made through the interpreter and the scheduler, respectively. The inputs are received through the interpreter, which filters and transforms the perceptions from the sensors into understandable units for further processing. The scheduler, the interface between the cognitive module and the effectors, organizes the actions according to priority and concurrence restrictions. Each emotional virtual character in COGNITIVA holds beliefs about the current state of the world, including defining characteristics, transitory states and attitudes, as well as knowledge of past events. The effects of perceptions on emotions and moods are based on the mechanism of expectation, while the effect of emotions on actuation is based on the mechanism of concern. An overview of the COGNITIVA architecture is presented in Figure 2.4.

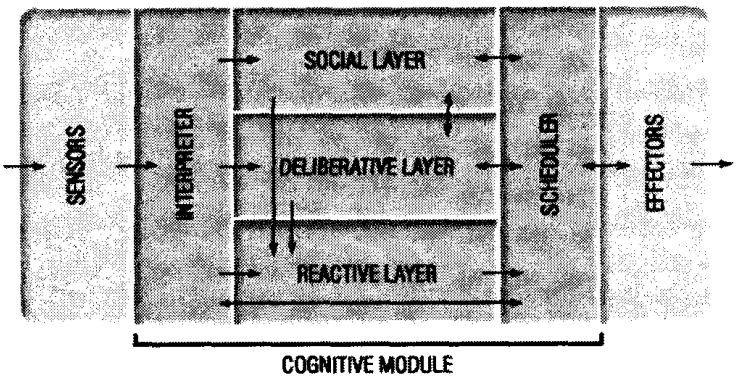

Figure 2.4: General structure of COGNITIVA (Imbert and Antonio, 2005a).

2.3.2.2. Personality Modeling

Personality in agents is referred to the characteristics of the individual consistent patterns of thinking, feeling and acting (Pervin et al., 2001). Trait theories that used traits as descriptors to describe personality was proposed by Gordon Allport (Allport, 1945), Raymond Cattell and Hans Eysenck (Hewstone et al., 2005). In trait theories, traits are labels given to represent the consistent aspects of personality and are viewed as continuous dimensions (Hewstone et al., 2005). Most trait theorists agree that all people have a fixed number of basic dimensions of personality (Xiao, 2005). Traits theories are widely used in computational models of personality, because the traits from everyday language are easy to understand and it is easy to convert the trait dimensions into a computational model (Xiao, 2005).

Rousseau and Hayes-Roth developed a social-psychological model for synthetic actors, in which they build their personality model on two types of theories: the trait theories and the social learning theories (Rousseau and Hayes-Roth, 1998). They used knowledge structures, for example, DEFAULT (<trait>, <value>), CORRELATED (<trait>, <component>, <corr-factor>) and T_SITUATION (<trait>, <priority>, <conds>, <value>) to model personality. Similarly, they modeled mood using the following knowledge structures: SELF_MOOD (<mood>, <value>), AGENT_MOOD (<mood>, <character>, <value>) and AG_SITUATION (<mood>, <type-situation>, <value>). Like the other components, ATTITUDE (<attitude>, <agent>, <value>) and

GROUP_ATT (<attitude>, <value>) are used to represent the attitudes towards other agents in the system. Finally in action selection, they assigned different personality traits and weights for different actions to be rated for competition. This model presented a social-psychological framework that connects personality traits, moods, and attitudes to allow synthetic actors to improvise their behaviours spontaneously without planning with respect to the directions they receive.

Rizzo and his colleagues proposed a personality-driven social behaviour architecture for believable agents, which models personality based on goals and plans instead of just varying the mapping between emotions and reactions (Rizzo et al., 1997). A personality could be defined as a set of characteristic and non-characteristic actions, but the problem is such actions are very many and can vary from one context to another. Representing personalities by a set of typical behaviours can be too expensive and lacks generality. As a solution, they defined personality as a cluster of goals with different priorities and a set of preferences over plans for achieving the goals.

2.3.2.3. Social Relationships and Interactions Modeling

The varied ways people interact with each other according to their relationships in society give us a deeper view on how personality, emotions, and belief influence our behaviour in the context of society for making a believable character (Baillie-De Byl, 2004). Adding relationships and interactions built on a social context can also give the player the illusion that NPCs have feelings that can be affected by others, and thus more immersion can come out of the gaming experience. From a sociology perspective, much of a human's daily behaviour is social, which means individuals are nearly continuously being influenced by the feelings, thoughts, and behaviours of others. Social psychology, the study of these influences and their reactions, has three

major concepts: attitudes, social cognition, and social influence (Lefton, 1982). Attitude is usually referred to as a general positive or negative feeling towards an object, person, or issue (Baillie-De Byl, 2004). Social cognition is the process of forming an impression of another's internal states and beliefs, deducing the motives and intentions from one's behaviours, and finally using one's own beliefs and behaviours to make sense of others. Social influence involves the ways that one's own beliefs and behaviours are affected by those of others. Social interactions give us a way to experience social influence and to shape and identify one's unique persona and characteristics.

Gruenwoldt and Katchabaw proposed a Realistic Reaction System (RRS) to model and maintain the relationships of the objects in the game world dynamically, providing the characters an ability to query the relationship network to react appropriately in behaviours and dialogues (Gruenwoldt et al., 2005a). Later on, Gruenwoldt and Katchabaw published a paper, in which they worked on the logic and mechanisms combining the RRS with AI controllers for reactive NPCs (Gruenwoldt et al., 2005b). In the relationship network, they built various relationships between the game entities, directed or undirected, depending on the relationship. The relationship types can have subtypes, presenting a hierarchical tree of relationship types to the users. For each relationship, attached with several attributes, they assigned origin, history, regularity, strength, polarity, and validity. Relationship-specific attributes can also be included where appropriate. Relationships get updated once entities got involved in the game events directly or indirectly through the propagation in the relationship network. The impact of time shift is also considered, by drifting the relationship towards a neutral state if there are no events or interactions that would otherwise intensify them. They also implemented a mechanism to augment the AI controllers with relationship data from the relationship network using state machines and rule-based systems. Finally a new reactive NPC AI architecture was developed for the extension of RRS, including a relationship system, a relationship manager, an AI controller and a personality and mood filter. The relationship system is to provide

a raw access to the relationship network to the relationship manager, which can then provide a specialized interface to the relationship data to the NPCs. The AI controller is responsible for decision-making for NPCs in the game, likely driven by a state machine or rule-based system. The personality and mood filters are used to provide the AI controller with a layer of customization and specialization to allow for more varied NPCs.

Guye-Vuilleme published a paper discussing the basic functional requirements for a general social reasoning mechanism and proposing a high-level architecture for believable social agents based on Roles, Norms, Values and Types (Guye-Vuilleme and Thalmann, 2000). Guye-Vuilleme identified that the functionalities that a social agent architecture should include are: a multi-behaviour architecture, a behaviour switching mechanism, acting according to the social environment, a basic cognitive mechanism, standard behaviours, and acting according to social values. Various useful concepts were introduced by Guye-Vuilleme, including Role, Norms, Values and World view. Role can be treated as a coherent set of standard behaviours, including world view. Various attributes, such as age or gender, professional status, and so on can be linked to a role. Moreover, several roles can be linked to one individual and switched successively depending on the social context. Guye-Vuilleme believes that grouping a set of coherent behaviours into roles can greatly facilitate reuse. Norms are described as guides for behaviours, and it makes sense to attach norms to roles since roles, after all, are a specific set of coherent behaviours. Values, constituting a personal moral system, involve the "right" action selection to pursue an ideal abstract state. It is easy to integrate values with roles because most of the roles involved with professions have a typical set of values. World view is also important for our typification and classification of social behaviour, which could be a helpful feature for specific situations, such as dilemmas. Each role contains standard behaviours, values and types; it inherits data from its parent roles in the role hierarchy; it is assigned to an agent with a weight, along with other roles; it can be associated to other roles, and it is activated by a switching mechanism.

Roles can be described using the following structures (Guye-Vuilleme and Thalmann, 2000):

$Role_i$= ($Name_i$,$StandardBehavioursList_i$, $ValuesList_i$, $TypesList_i$)

$StandardBehaviour_i$= ($Name_i$, $PreconditionsList_i$, $ActionList_i$, $PostconditionsList_i$)

$Value_i$= ($Name_i$, $Weight_i$) where $0 <= Weight_i <= 1$

$Score_{ij} = f(Value_i, StandardBehaviour_j)$ where $-1 <= Score_{ij} <= 1$

$DefaultScore_{ij} = f(Value_i, Type_j)$ where $-1 <= DefaultScore_{ij} <= 1$

Roles are organized in a hierarchical way such that child roles can have multiple parent roles and inherit their parents' characteristics (Guye-Vuilleme and Thalmann, 2000). Multiple roles can be signed to an agent with a weight allowing for specifying their relative importance to the agent's personality. Roles can also be connected to other roles on a logical relationship, such as a symmetrical relationship of father and mother. The role-switching mechanism can be implemented using activation energy which comes from the initial role weight, the location, the time and the interactional context. Guye-Vuilleme believes that this role-based organization allows the emergence of rich and surprising behaviours as more and more roles get involved, and produces interesting interactions between conflicting values and norms.

Sweetser also talked about artificial social networks with respect to emergence; she believes that by introducing the power of social dynamics into games, we can potentially add more life and interactions to game worlds (Sweetser, 2008). Sweetser theorized that artificial social networks can be formed in games by attaching characters to status, social connections, memory of other characters, and attitudes toward characters and the player. Emergent social interactions and character behaviours can be formed when we use these social networks to manage the flow of information in game worlds and character behaviours in social situations. She also modeled opinion in social networks, in which each character maintains a set of variables representing their opinions, as well as links to other characters with weights.

Characters' opinions change over time according to the weighted sum of the opinions from those connected. In this way, if a character feels strongly negative toward the player while others do not, the character's opinion will drift toward being more positive over time. If weight between characters is positive, their opinions will align over time, otherwise their opinions will drift apart. Therefore, polarized weights can be used to identify whether or not the character is a friend. Moreover, the connection weight is subject to change under the following circumstances: a positive or negative interaction occurs or a game event affects the relationship. Finally the change of opinion would be the weight sum of the difference in opinion of all the connected characters modified by the connection strengths. Opinion flow in the social network, interactions with the player, as well as other game events, can all change characters' opinions.

Another system that emulates the opinion flow is Alt's dynamic reputation system based on event knowledge (Alt and King, 2002). In *Ultimate Online*'s reputation system, killing an NPC changes your karma and fame, which influences all other NPCs' opinions of you instantly, whether or not they actually witnessed the event. In order to solve this extrasensory perception (ESP) problem, Alt proposed a reputation system tying changes in opinion to direct or indirect knowledge of event (Alt and King, 2002). The main data structures used are the Event Template, the Reputation Event, the Master Event List, the Per-NPC Long-Term Memory, and the Per-NPC Reputation Table. A Reputation Event, which contains information of who did what to whom, as well as the reputation effect, is triggered by the reputation system when an action that might change the player's reputation is performed. The event is built on a static Event Template, and uses dynamically generated information to fill in the blanks. The Reputation Event is then added into the Master Event List, and is announced to all witnessing NPCs. Each NPC maintains a Long-term Memory, which is a list of references to the events in the Master Event List that the NPC directly saw or indirectly heard about. Each NPC stores its opinions of the player, as well as of other groups in a Reputation Table, which will be updated as a Reputation Event

occurs. The NPCs can learn about events from Event Announcers, as well as from sharing information with other NPCs, and then perform the memory match and update process.

In Alt's dynamic reputation system, NPCs in the same group can only hold the same opinions about other groups of people (Alt and King, 2002). This is simply not realistic enough for general use. Therefore, Darken (Darken and Kelly, 2008) introduced a technique that enables individualized NPC attitudes with social networks based on Kelly's work (Kelly, 2007). Darken described his mathematical attitude model to us: $\mathbf{x}(n+1)=\mathbf{A}\mathbf{x}(n)+\mathbf{B}\mathbf{u}$, where $\mathbf{x}(0)=\mathbf{B}\mathbf{u}$.

A: Matrix of affinities.

$A_{i,j}$: The affinity of NPC i for NPC j. $A_{i,j}=0$ for all $i=j$ as we handle self-affinity below.

B: Diagonal matrix of self-affinities

$B_{i,j}$: The self-affinity of NPC i, $B_{i,j}=1$ if $i=j$, and zero otherwise.

u: The vector of direct effects.

$\mathbf{x}(n)$: The attitude change after n updates.

The constraints are that for all i, $B_{i,j}>0$ and $\sum_j |A_{i,j}|<1$.

This method helped automate NPC changes in attitude due to an action (Darken and Kelly, 2008). Kelly pointed out that **A** can be an affinity matrix of multiple networks defined by developers; examples of these networks could be family connections, networks of friends, coworkers, and so on (Kelly, 2007). Regarding the problem of when to stop the iteration, Kelly suggested that we solve x by $x=Ax+Bu$, $(I-A)\,x=Bu$, $x=(I-A)^{-1}Bu$.

Another instance of implementing opinion systems in a game is *Fable* from Lionhead Studios. In this game, instead of NPCs holding opinions of each other, it is rather about the player holding social reputations of NPCs for the construction of his/her own social persona (Russell, 2006).

2.3.2.4. Action Selection Mechanism

In the design of a game, one of the key issues is how to select appropriate actions to exhibit goal-oriented behaviour when given many possibly conflicting goals (Pinto and Alvares, 2006). Behaviour networks were proposed as a mechanism to select good enough actions in complex and dynamic environments (Pinto and Alvares, 2006). Later, extended behaviour networks were introduced, which can be viewed as a set of linked modules and goals that mutually inhibit and excite each other through activation-spreading, flowing from goals to modules. The modules with higher activation, excitability and fewer resources are selected for execution at each step. Pinto and Alvares have successfully built a game agent with goal-oriented behaviour and simple personality using extended behaviour networks.

Mac Namee and Cunningham developed the μ-SIC system for creating socially interactive NPCs, which uses psychological models to simulate characters' personalities, moods and relationships (Mac Namee and Cunningham, 2003). The values of these models are fed into an Artificial Neural Network (ANN) to produce a resulting behaviour. The ANN used in the μ-SIC system is used to generalize smoothly from a set of handcrafted examples to cover the entire space of possibilities.

2.3.2.5. A Framework for Believable NPCs

A framework for realistic psychosocial behaviour in NPCs, developed by Bailey, uses concepts behind emergent gameplay to support the mechanics of enabling designer-defined psychosocial concepts, undefined circumstances, and emergence (Bailey, 2007). Bailey also implemented a prototype based on the framework. Bailey's framework will be examined in further detail later in this thesis.

2.4. Discussion

Reactive, triggering, goal-based, utility-based, and anytime agent architectures were generic agent architectures discussed in (Baillie-De Byl, 2004). However, except for utility-based architectures, the rest were not able to support emotional simulation. Baillie-De Byl also talked about Soar, H-CogAff, EXCALIBUR, and being-in-the-world architectures in (Baillie-De Byl, 2004). Since the hybrid agent architectures were developed from the generic ones, they could support emotional simulation, but none of these architectures seemed to support social behaviours.

The Oz project supports emotion, personality, social relationship and aspects of self motivation; however, it provides no support for character growth and change (Bates et al., 1994). Moreover, the social relationship in the Oz project is simply defended as like and dislike, which omits many conflicting social relationships.

Paanakker' work provides insights as to how to integrate the emotion component into the traditional sensing/thinking/acting agent architecture and how emotion can affect our perception and thinking processes (Paanakker, 2008). Furthermore, the emotion activation functions inspire us on how to create characters with a rich set of subtle differences.

Zhou proposed an interesting emotion transition model for NPCs based on the PAD emotional state model in his paper (Zhou et al., 2006). His model takes the emotional influence from the environment into consideration. However, personality and social ties are not included in Zhou's model.

COGNITIVA (Imbert and Antonio, 2005a, 2005b; de Freitas et al., 2007) is an abstract and generic, domain-independent architecture that allows for multiple theories; therefore it is applicable to a wide range of domains and problems. However, the performance of such an architecture was not mentioned when applied in a real-time

interactive game.

The social-psychological framework for synthetic actors developed by Rousseau and Hayes-Roth has included social-psychological components such as personality traits, moods, and attitudes to allow for improvisational behaviours (Rousseau and Hayes-Roth, 1998). However, the framework not only did not mention how to update the values of moods and attitudes, but also ignored the influence of emotion, social attitudes and goals on action selection. Therefore, it can not produce a satisfactory believable social-psychological character.

The personality model based on goals and plans, proposed by Rizzo and his colleagues (Rizzo et al., 1997), is a creative idea without the need to list the typical behaviours exhaustively. However, in reality, personality only plays a small part in the decision of goals and plans. Emotions, social context, belief, value, and human needs are also need to be taken into consideration.

Gruenwoldt and Katchabaw's RRS (Gruenwoldt et al., 2005a) and the extension of RRS (Gruenwoldt et al., 2005b) provide us with an inspiration on how to combine a relationship network, personality and mood filter together for the AI controller to make realistic decisions in a social context.

Guye-Vuilleme (Guye-Vuilleme and Thalmann, 2000) introduced many important concepts: roles, norms, values and types, which are very helpful to organize and group agent attributes and behaviours in a social context. Furthermore, the concepts of role, type, and role hierarchy help make a variety resources in a game reusable.

Sweetser discussed how opinion flow travels in a social network, but she did not talk about how an event's influence on opinions gets propagated in the network (Sweetser, 2008). Alt developed a dynamic reputation system that can change an NPC's opinion based on the direct or indirect knowledge of the event (Alt and King, 2002). However,

NPCs in the same faction can only share the same opinion about an entire group of people; to improve that, a technique that enables individualized NPC attitudes with social networks with continuous updates was introduced by Darken (Darken and Kelly, 2008). However, Kelly admitted that the main difficulty for this approach was that it required a large initial investment of time and effort during development instead of simply assigning a faction and walking away (Kelly, 2007). This approach requires the developer to take the time to develop at least a basic background story for each and every agent in the simulation, which is a difficult task for developers as the number of agent grows unless tools are developed beforehand. This kind of opinion system was implemented in the game *Fable* (Russell, 2006), which is an acceptable simplified implementation of social relationships in a real-time interactive video game.

Pinto and Alvares's extended behaviour networks (Pinto and Alvares, 2006) is only limited to mapping from a few variables to a limited number of actions; however, when it comes to choosing an appropriate action based on one's personality, emotional states, social context, and other internal states, it might get out of hand quickly. Using ANN like what Mac Namee and Cunningham did in their μ-SIC system is fine in theory, but is not practical, however, for training the ANN with a large quantity data in a complex believable NPC real-time system (Mac Namee and Cunningham, 2003).

Bailey's framework (Bailey, 2007) for realistic psychosocial behaviour in NPCs uses a creative emergent technique to model psychosocial behaviour, which is capable of dealing with undefined circumstances and allowing for improvisation. Furthermore, her prototype showed encouraging results during evaluation. However, she did not extensively explore the details of psychosocial models used. Hence, it is left to others to seek the way to integrate the popular psychosocial component models into her framework and make it useable to game developers.

The current work will support the integration of customizable emotion, personality models, social roles and other psychosocial concepts to exhibit believable emergent social behaviours while allowing the flexibility in terms of scalability in models and data-driven in system.

Chapter 3

3. Design

The core design for this believable NPCs architecture is through modeling and integrating various psychosocial components, and emergence will to be used for run-time system development. As mentioned before, focusing on a bottom-up view of building the game world, emergence relies on simple component-level behaviours (or stimulus responses) interacting with each other to generate complex system-level behaviour. This could be achieved by creating objects which are self-centered, goal-directed, and responsible for their own needs and responses (Bailey, 2007). In a psychosocial context, those objects could be emotion traits, personality traits, social ties, NPCs and other psychosocial components.

We first examine the overall approach in this work in Section 3.1. Section 3.2 describes the NPC data models design, followed by Section 3.3 examining the design of robust and flexible NPCs creation and management tools. In Section 3.4, we discuss the use of emergence in building our framework in more detail, and present a conceptual design of our work.

3.1. NPC Architecture Design Overview

Our NPC architecture design consists of three parts: NPCs creation and management tools, data models and NPCs data, and a run-time system. NPCs creation and management tools are responsible for composing, generating, and maintaining NPCs data and the data models, which defines the NPC from personality, emotions, social information and message perspectives. Role, as a concept to represent social information, can include the associated typical goals and actions. The data models and NPCs data are then sent to the run-time system as input to generate output behaviours

for NPCs. There are three modules in the run-time system: the Data Processing module, the Emergent Framework module and the Rendering module. The Data Processing module is used to generate structured data from raw data to make it useable within the Emergent Framework module, which does all the computation and outputs behaviours for rendering. Note that our architecture design supports all different kinds of emergent frameworks; we used Bailey's emergent framework in our design for convenience. However, Bailey's emergent framework is incomplete for our purposes in this thesis and we needed to modify it and add modules to it to make it more robust, flexible, comprehensive, and able to support the data models. We will also discuss the key elements of Bailey's emergent framework that are relevant to our work in Section 3.4 to make it clear how the system as a whole functions and how its various elements interact with one another.

The overview design of our NPC architecture is shown in Figure 3.1.

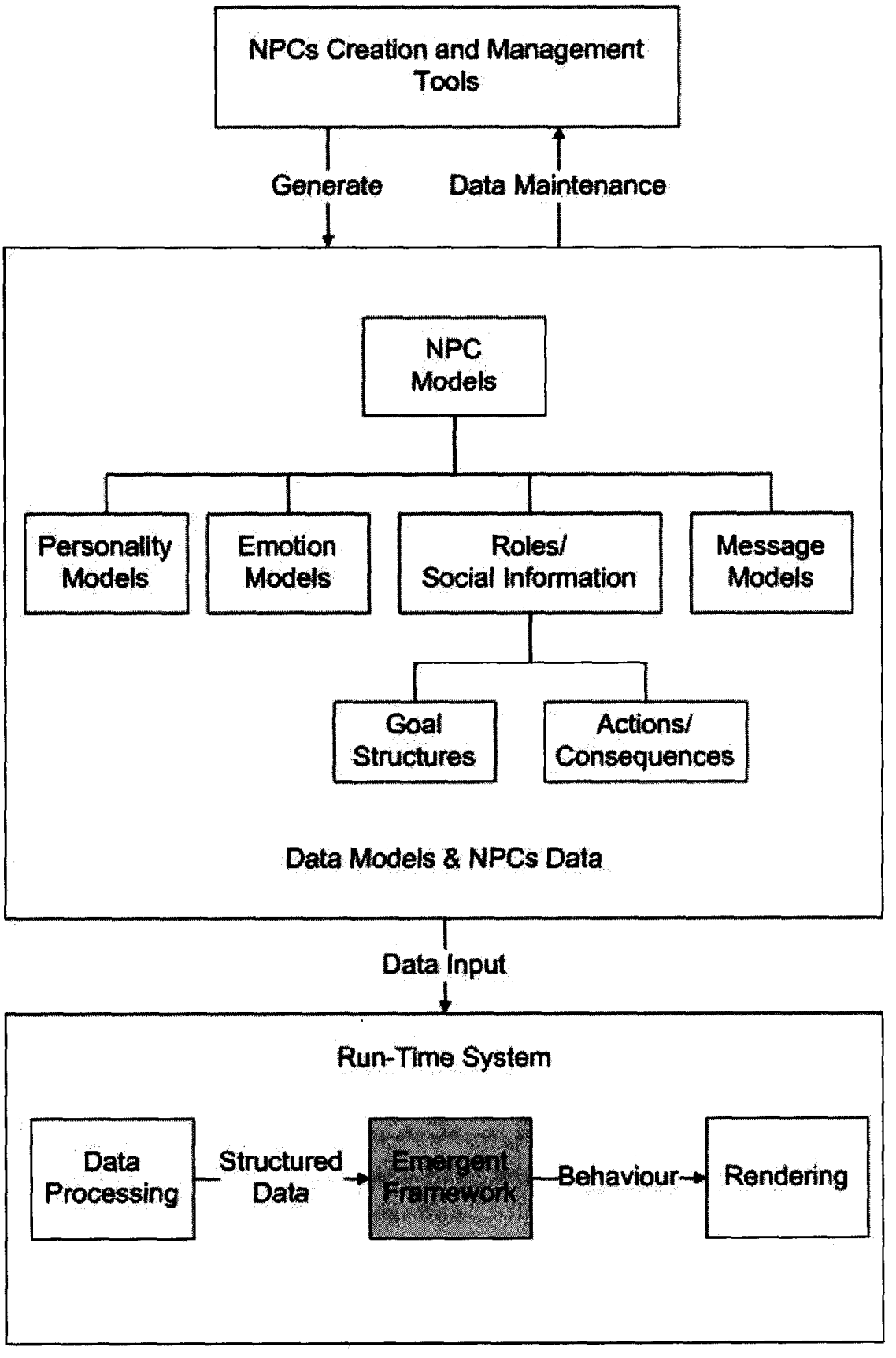

Figure 3.1: An overview of NPC architecture design.

In the next section, we discuss the model design of NPCs, followed by a discussion of the NPCs creation and management tools design. Finally, we examine the emergent

framework design in the run-time system.

3.2. NPC Data Modeling

In this section, a detailed low-level design of the NPC data is discussed, focusing on how to model each psychosocial component to fill in the middle module presented in the centre of Figure 3.1.

3.2.1. Elements of a Believable NPC

To construct a believable NPC, many attributes need to be assigned to direct its behaviours, including emotions, personality, social ties, goals, and other psychosocial components.

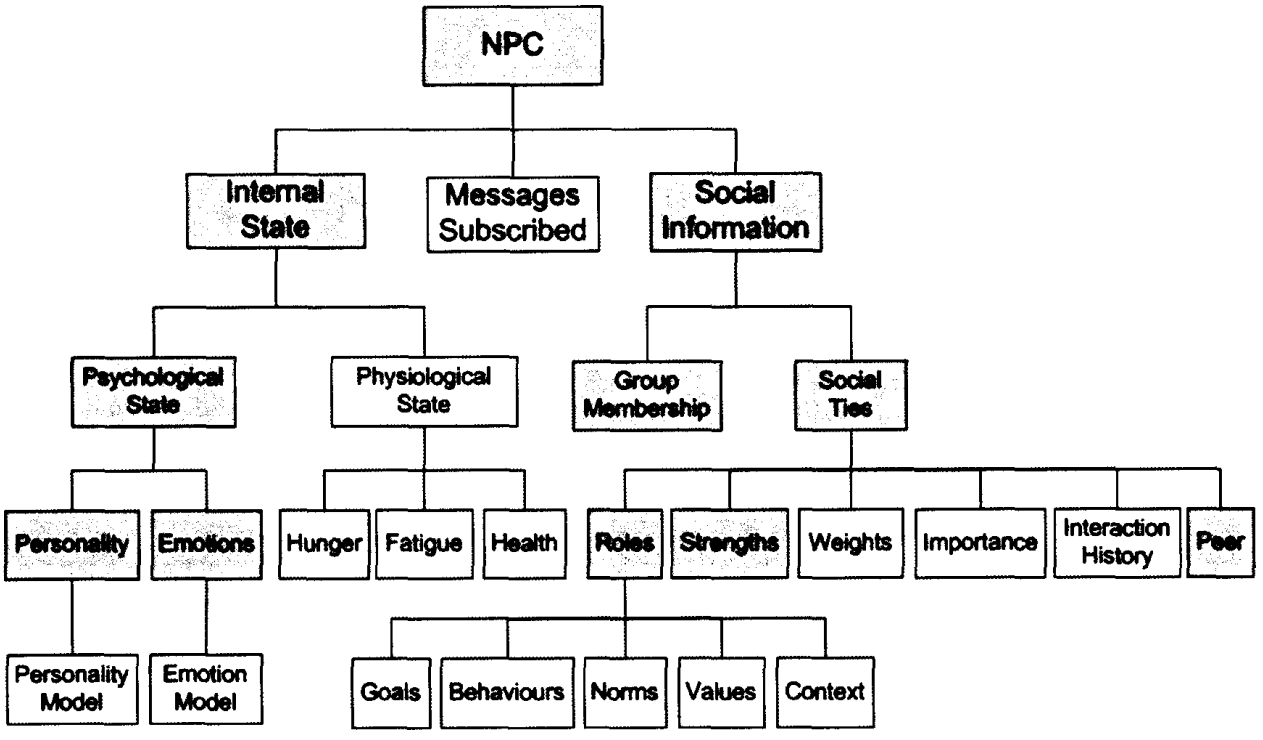

Figure 3.2: A sample of an NPC psychosociology.

Figure 3.2 depicts a sample typical set of psychosocial attributes for believability, including internal state (a psychological state and a physiological state), social information (social ties and group membership) and the social event messages that NPCs are concerned with. The coloured modules in Figure 3.2 are listed in Bailey's framework (Bailey, 2007), however it needs extensions and detail refinements to fill in the various modules.

Note that not all of the attributes listed here are necessary for building a believable NPC architecture, nor is this set of attributes complete. Different attributes may be added or removed depending on the requirements of the game in development. However, this is a typical set of elements; it is believed that some form of psychological state and social ties, at a minimum, should be included to create a believable character (Bailey, 2007).

The following sections will discuss some of the considerations in developing an NPC's internal state (3.2.1.1) and social information (3.2.1.2).

3.2.1.1. Internal States

The internal state of an NPC could be roughly split into psychological state and physiological state. A typical example of some internal states that one might consider is to include a model of emotion and a personality model, as discussed later in this chapter. However, there are various personality models and emotion models focusing on different areas. Furthermore, different games have different requirements from their psychological models. Consequently, the believable NPC architecture should be compatible to various famous personality models and emotion models that have already developed by the theorists and allow for the game designers to customize their own models. To enable this flexibility and customization, these attributes should be stored as data, instead of hard-coded into the software.

As for physiological state, some attributes to consider are hunger, fatigue and health. Those attributes in games are usually modeled as some value bars indicating the physical status of the character, for example, its life bar. Another use of these attributes in games is to treat these physiological states as self-motivation factors or basic goals for the NPC, as these attributes account for the primary needs in Maslow's

hierarchy of needs model.

3.2.1.2. Social Information

Social relationships can be handled by having the NPC maintain social ties to others and group memberships to the groups the NPC belongs to. Each social tie can include a set of roles, a set of strengths, a set of weights, an importance value, a group of interaction history and a pointer to the peer in the relationship. A role is a coherent set of standard behaviours, which contains a typical set of goals, a set of actions (behaviours), a set of norms (guides for behaviours), a set of values (moral system), and a context for role activation. For each role in the social tie, it associates a strength value of the relationship to the peer, and a weight of the current role. The importance value is used to indicate how important the social tie is and the interaction history records the memory of the events that have occurred between the NPC and the peer. The pointer to the peer can help to indentify the social tie, as well as to allow for responding directly to the peer. The social ties can also be modeled to be either symmetric or asymmetric (such that two people may feel different about each other, rather than sharing the same feelings mutually). Group memberships can be modeled as a list of groups that the NPC belongs to. Further details related to the modeling of social information are provided in Section 3.2.6 and Section 3.2.7.

This section has outlined the elements important to the believable NPC architecture. The next sections will discuss the important elements outlined in more detail separately.

3.2.2. Personality

Currently, there are many personality models that are widely used, and the three leading models according to literature citations are the Myers-Briggs Typology, the

Five Factor Model (FFM or OCEAN) and Reiss' personality model. As discussed earlier, different personality models have different focuses, and there are some overlapping personality traits between them. Unfortunately, there is no single comprehensive personality model that can be directly applied to any game, since different games have different requirements on personality traits that the developers may want to use. For example, a game designer may wish to use a concise personality model for simplicity, rather than a more comprehensive model. A game designer may wish to use Myers-Briggs Typology and Reiss' personality model at the same time in different aspects in the game as well. Since no single model can comprehensively cover every personality trait that a developer may need, and, at times, the leading personality models are too complex and impose too much overhead for developing small games, using only one particular model throughout an entire game can be rather limiting. As a result, we bring together elements from multiple models at once to create an integrated, multi-model, multi-level, scalable and customizable personality model, which is unique and not seen in the literature in this area.

For instance, as shown in Figure 3.3, with our personality model, we could model a scientist (a typical profession with INTJ traits values in Myers-Briggs Type Indicator (MBTI)) who is generous (evaluated using the Agreeableness trait in FFM) and curious (evaluated using the Curiosity trait in Reiss' 16 basic desires), which is difficult to model using any current individual personality model. At the same time, our model allows for customization, which is to say the designers can get rid of the personality traits that are unnecessary for the games they are making for simplicity and efficiency. Alternatively, they can add new individual traits or complete models to suit their needs for additional expressiveness and robustness.

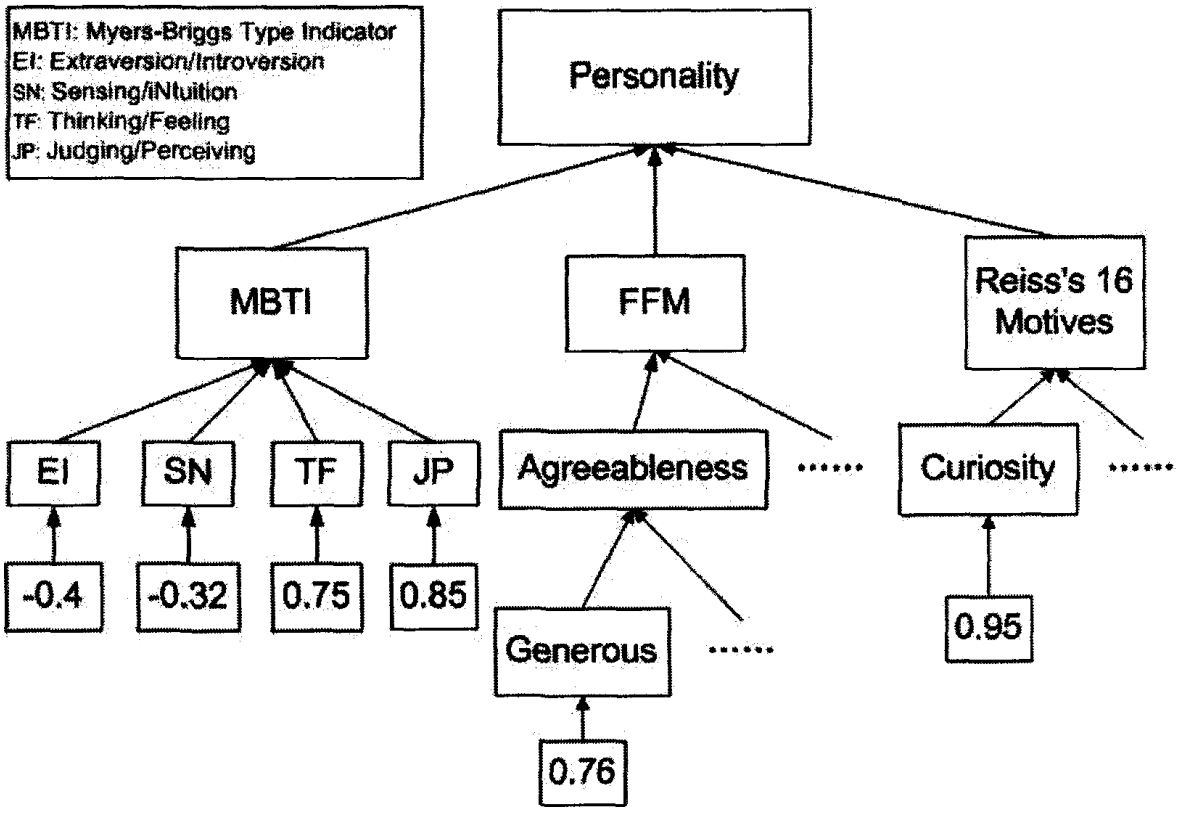

Figure 3.3: A sample personality model used to model a scientist NPC.

Our personality model is a two-dimensional model that consists of multiple personality levels, and each personality level is filled with personality traits. Each trait has a value ranging from -1.0 to 1.0, with -1.0 indicating that the trait is strongly negative in that respect, and with 1.0 indicating strongly positive. A value of 0 indicates a neutral stance with respect to that trait. For example, from Figure 3.3, a value of 0.95 in curiosity indicates that the individual is highly curious. A value of -0.95 would indicate that they are strongly not a curious individual. A value of 0 would indicate that the individual is neither strongly curious nor strongly not curious.

Personality traits in different levels can have links to each other, which have a value indicating the weight to/from that specific personality trait. Usually the higher the level the personality trait is in, the more abstract the personality trait will be. For example, as indicated in Figure 3.4, we might place the Extraversion/Introversion personality trait on the first level, and then followed by an Outgoing personality trait and a Bold personality trait on the second level, which contribute the weight of 0.7

and 0.3 respectively to Extraversion/Introversion.

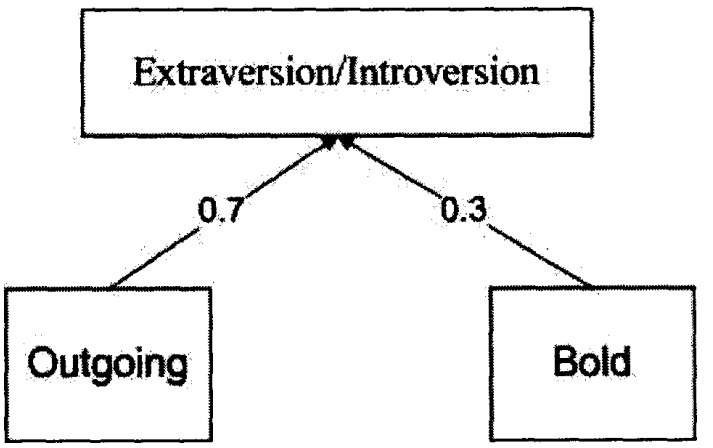

Figure 3.4: An illustration of the personality model.

One of the benefits of using this mechanism to build the personality model is that we can integrate all the prevalent personality models together, as many as needed, and scale our system very easily in a horizontal way by linking lower-level traits shared by different models together into their higher-level traits for applications in various game development environments. Figure 3.5 gives an example of scaling the personality model in a horizontal way by linking different personality models for different purposes into one. Later, Figure 3.7 shows the combination of traits across different models at the same time.

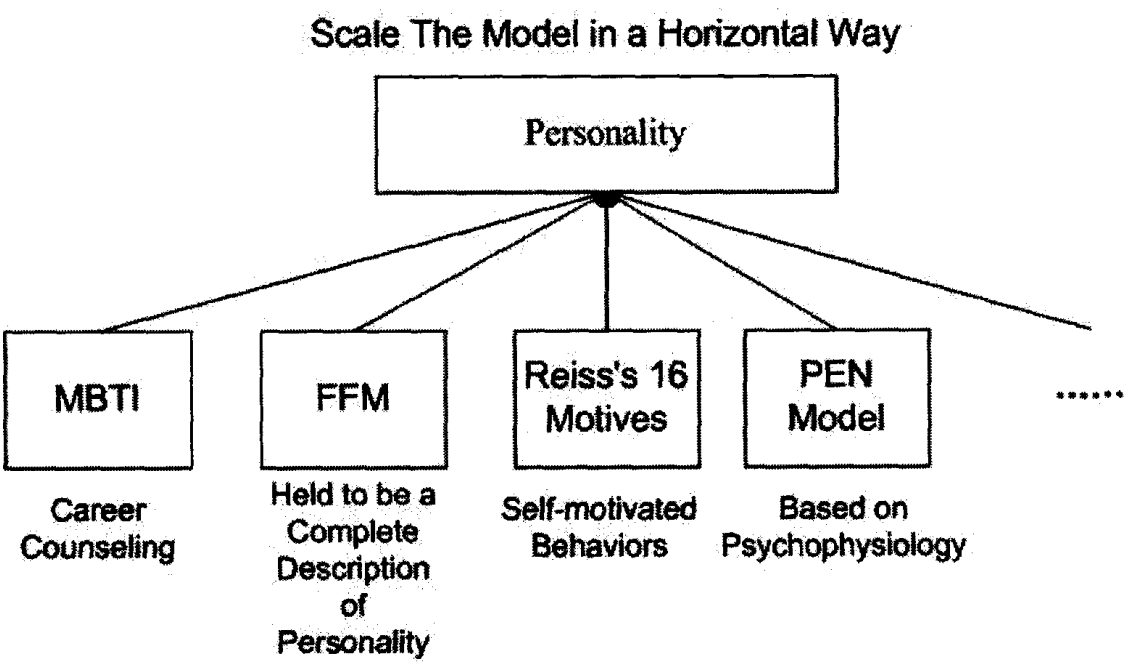

Figure 3.5: A sample of scaling the personality model in a horizontal way.

Another significant benefit of our personality model is that it scales well in a vertical way as well, and thus can support different levels of detail. As mentioned before, the five personality factors in FFM came from Cattell's 16 PF, and FFM can be subdivided into the traits from Cattell's 16 PF model in lower-level (Syque, 2002-2008; Sinclair), which can increase the accuracy and add another layer of emergence to games. For example, Figure 3.6 shows how to use the personality descriptors from Cattell's 16 PF to describe the trait Consciousness from FFM. Other than Cattell's 16 PF model, FFM also correlates with the occupational personality questionnaire model (OPQ) and the Belbin team role types, which also give us the possibilities to scale our personality model for versatile use.

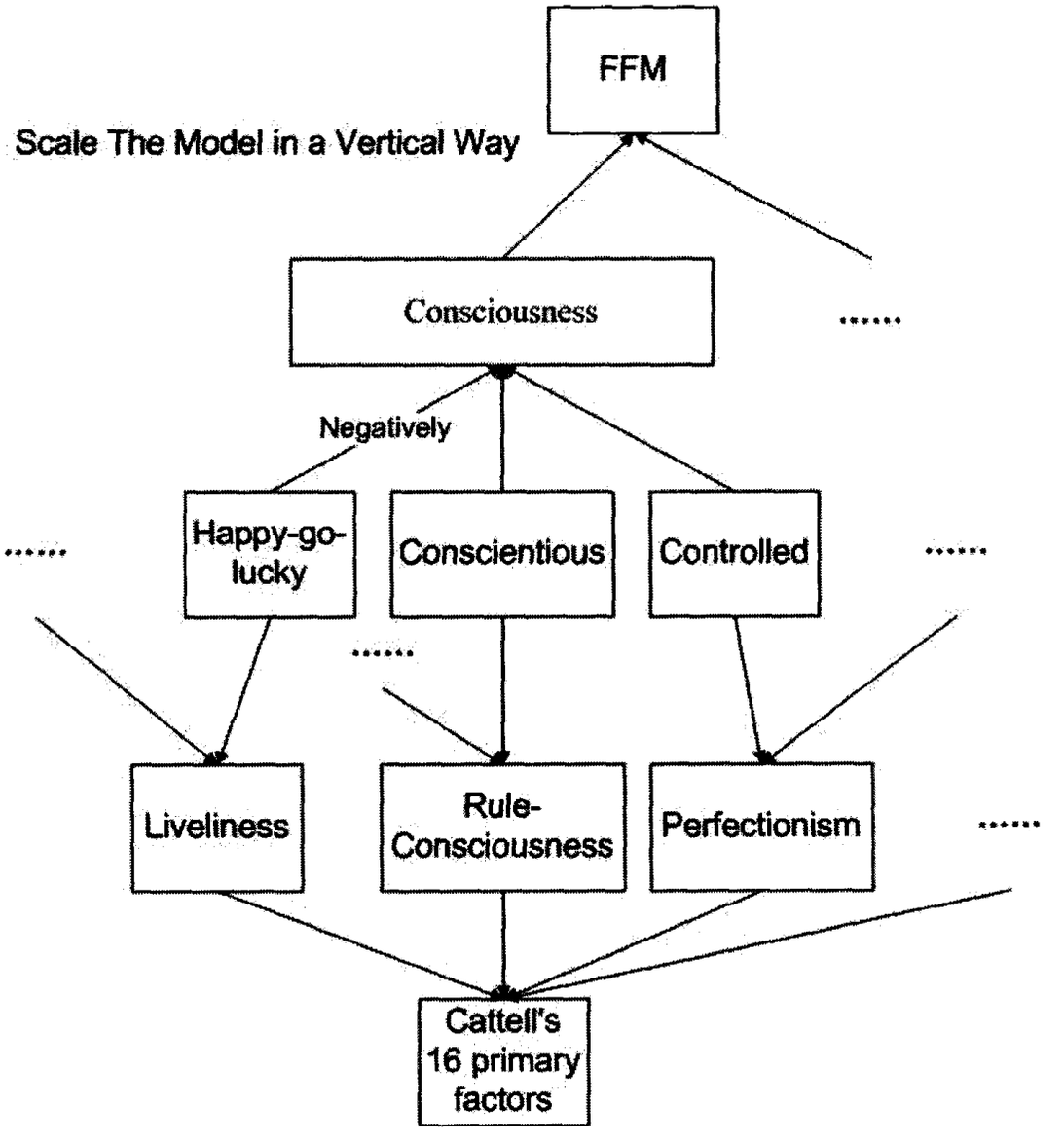

Figure 3.6: A sample of scaling the personality model in a vertical way.

It is important to reiterate here that there are overlapping personality traits between different personality models. A well known example from Myers-Briggs Typology and FFM is that EI and SN are strongly correlated to extraversion and openness respectively, while TF and JP are moderately related to agreeableness and conscientiousness respectively (McCrae and Costa, 1989). In order to avoid the conflicting case where EI in MBTI indicates the NPC is highly extroverted while Extraversion in FFM indicates the NPC is highly introverted when both of them are used in the user's design, we add another lower-level of personality with personality traits inside and use these lower-level traits to describe the two higher-level personality traits respectively. For example, we can simply add a lower-layer set of traits consisting of Expressive, Energetic, Outgoing, Bold and Talkative to describe EI and Extraversion respectively. Note that the traits Outgoing and Bold compose the main part of both EI and Extraversion, so that EI and Extraversion would not have contradicting values in representing similar personality traits. Figure 3.7 provides an illustration of this.

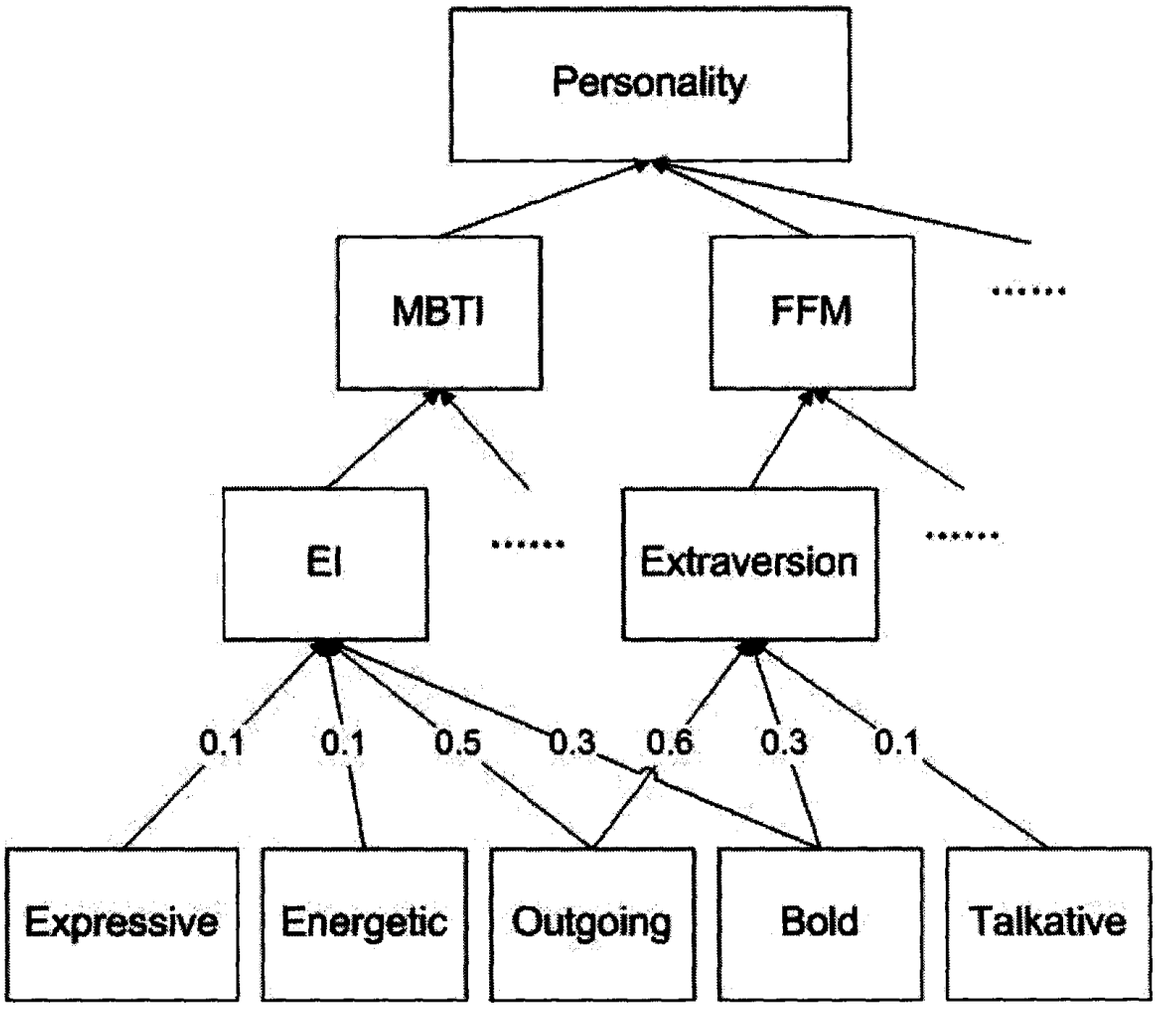

Figure 3.7: A demonstration of avoiding conflict between personality traits.

At the same time, when multiple personality models with overlap are used at the same time to describe NPCs, to represent the subtle differences among the similar personality traits from different models, we can simply use different lower-level personality traits linking to the higher-level ones with different weights. As a demonstration, SN in MBTI and Openness in FFM are highly correlated. To identify and highlight the subtle differences between them, we add a set of traits including Idealistic, Abstract, Imaginative, Deep and Creative as lower-level traits. By using Creative exclusively to represent Openness and using Idealistic and Abstract exclusively to represent SN, we can distinguish the subtle differences between them, as well as by assigning different weights to the common traits Imaginative and Deep. This approach is shown in Figure 3.8.

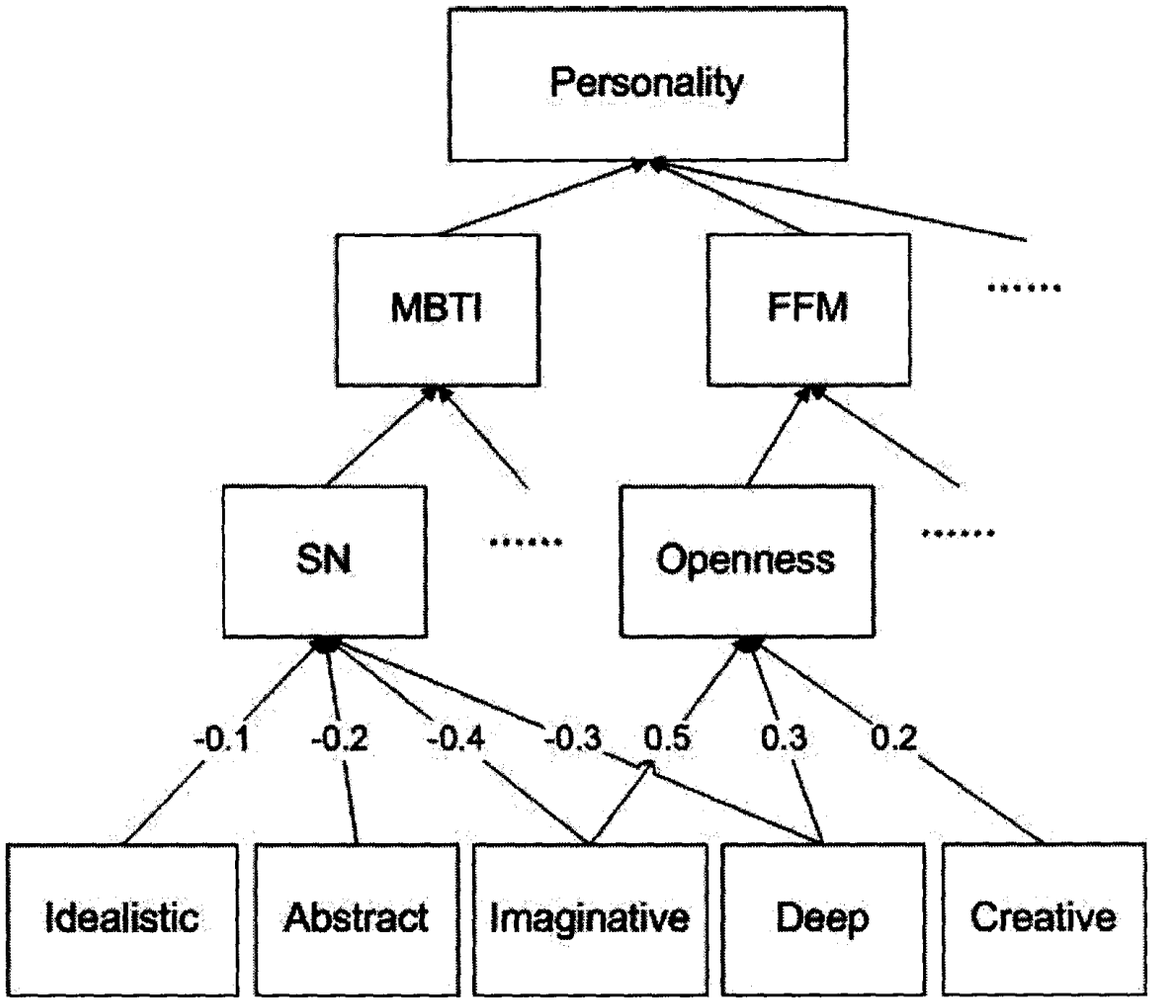

Figure 3.8: A demonstration of representing subtle differences between personality traits.

Again, the correlations between different personality models provide us the basis to expand and scale our models in both horizontal and vertical ways.

3.2.3. Emotion

As with personality models, there are many different emotion models from different theorists. Among these emotion models, the famous ones include Plutchik's basic and advanced emotions model, Parrott's tree structure list of emotions and Ekman's universal facial expression emotions.

Again, the various emotion models are built for different purposes. Arnold's model is

for action tendencies; Frijda's model is for action readiness; James' model is for bodily involvement; McDougall's mode is for relation to instincts; Mowrer's model is for unlearned emotional states; Plutchik's model is for relation to adaptive biological processes; and Tomkins' model is for density of neural firing.

For the same reason, using a specific emotion model is not robust or expressive enough for being applied in different games. Therefore, as we did for personality in the previous section, a scalable, customizable two-dimensional multi-level model, which involves elements from different existing emotional models, will be developed as our emotional model.

The same way as we used to model personality, emotion model traits at lower-levels are linked to and referred to by traits at higher-levels, tying the emotion models together. With this kind of integrated emotion model, we can use Ekman's model for our NPCs' facial animation control, Plutchik's model to simulate advanced emotions, and Parrott's hierarchical emotion model for subtle emotion changes all at the same time. For example, Ekman's model includes the emotion traits disgust and joy, and both of them are among the basic emotion traits in Plutchik's model to form the advanced emotion trait love. (To be more precise, love consists of joy and acceptance, while disgust is the opposite of acceptance in Plutchik's model.) Parrott's hierarchical emotion model also includes love as a primary emotion, disgust as a secondary emotion described by tertiary emotions contempt and others, and joy as a primary emotion described by secondary emotions, one of which is optimism that can be further divided into tertiary emotions hope and others. An illustration of this is in Figure 3.9. Note that even we draw on the same emotion traits data shared among different emotion models, the models do not influence each other.

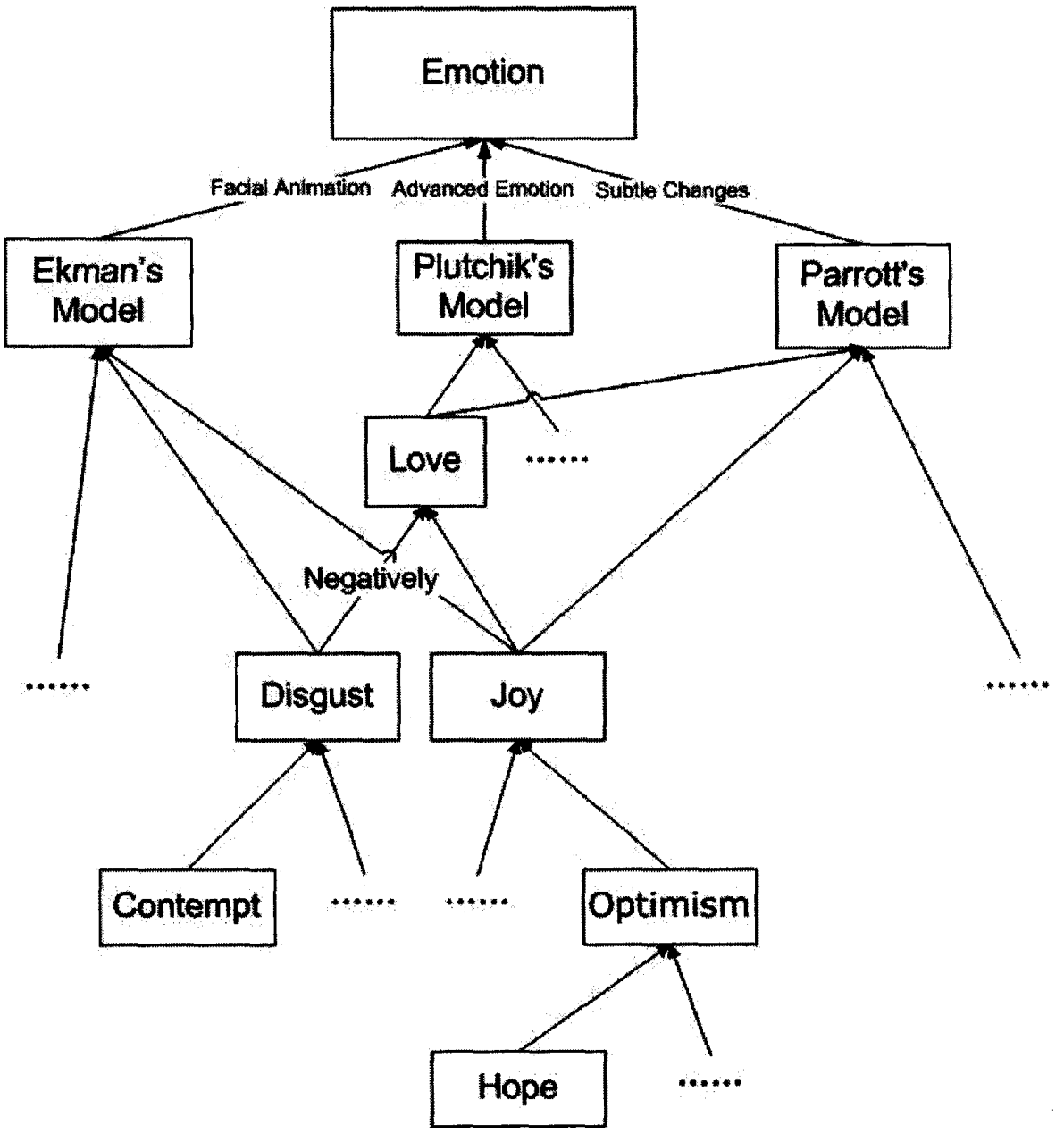

Figure 3.9: An Example of using our integrated emotion model for different purposes.

Likewise, in our emotion model, each level is filled with emotion traits which take values ranging between -1.0 and 1.0 much like personality traits as described in the previous section. Emotion traits in one level can also have links to emotion traits in other levels. Each link will be associated with a weight indicating the percentage of the value contributed to another emotion trait. Normally, the more abstract the emotion trait is, the higher the level we will place it in. As a demonstration in Figure 3.10, we can place the advanced emotion Love from Plutchik's model onto the first level, and then we can put the basic emotions Joy and Acceptance onto the second

level with weights of 0.6 and 0.4 respectively to Love.

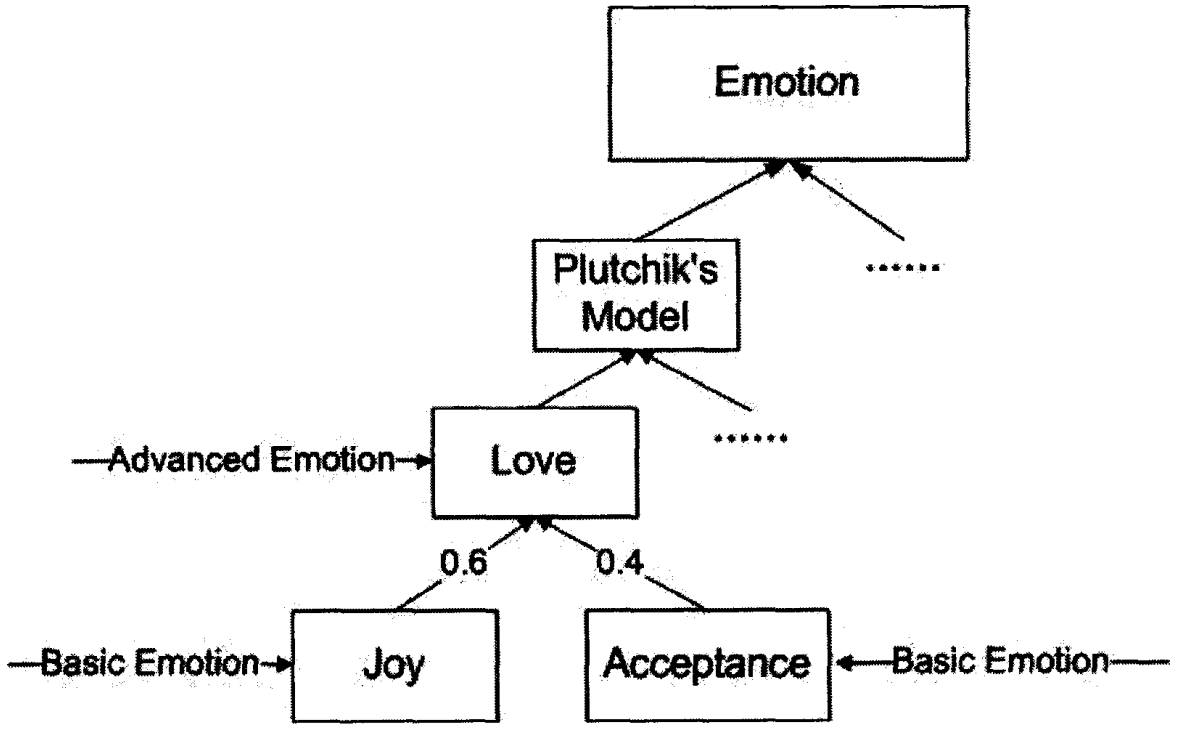

Figure 3.10: An illustration of the emotion model.

Modeling emotion in this fashion can bring us the advantages of being easy to scale the model in both horizontal and vertical ways. By integrating the various different emotion models that we need, we can make use of different models in their specialized areas and also make our emotion model applicable to various game development scenarios (see Figure 3.11 for an example).

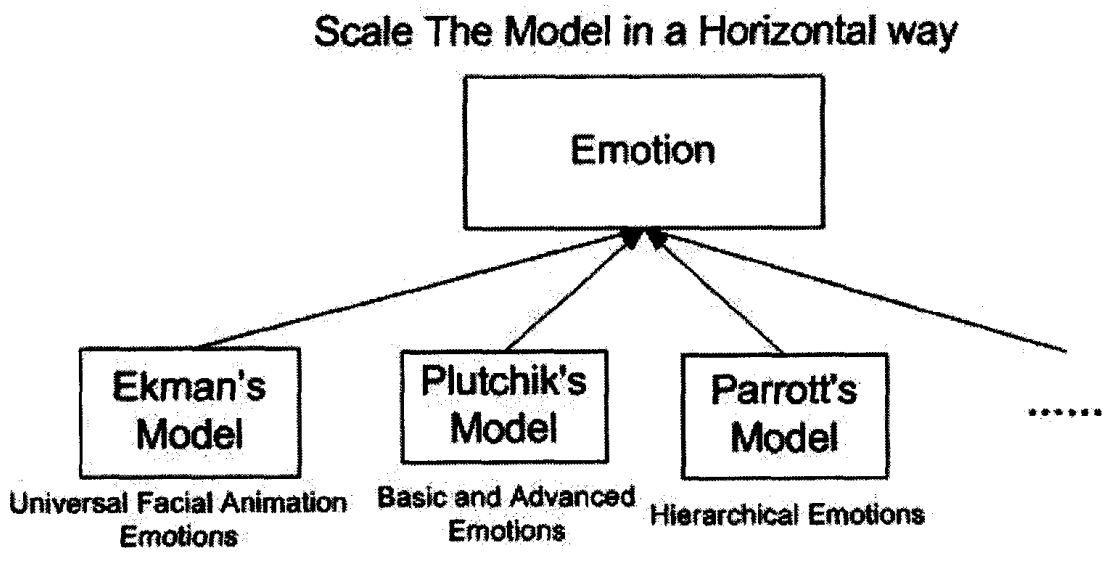

Figure 3.11: A sample of scaling the emotion model in a horizontal way.

By placing the Primary emotions, Secondary emotions and Tertiary emotions from Parrott's categorized emotion list into different emotion levels of our model, or putting advanced emotions and basic emotions from Plutchik's model into different emotion levels of our model, different levels of detail of emotion can be offered to the user (Figure 3.12).

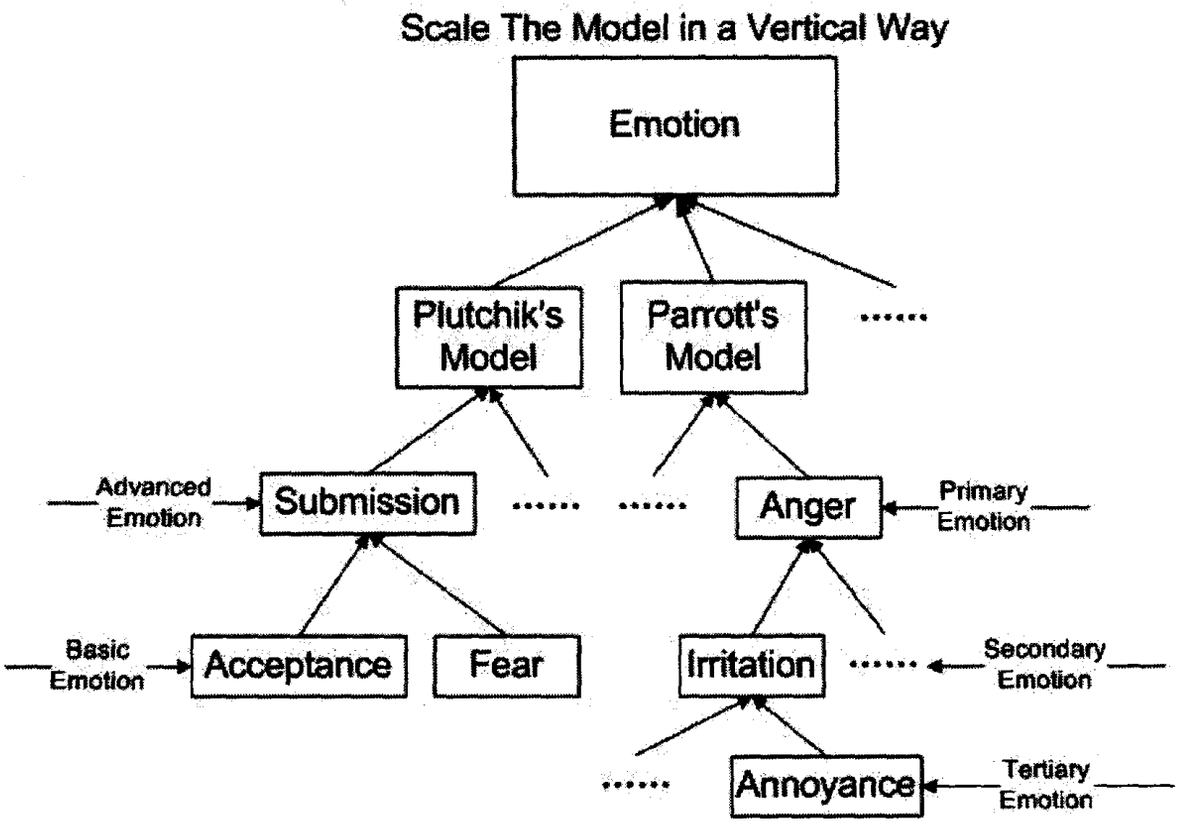

Figure 3.12: A sample of scaling the emotion model in a vertical way.

For the overlap between two emotion models, if the two emotion traits are not exactly the same, we can use the same lower-level emotion traits with slightly different weight contributions to describe them respectively as we did for our personality model to avoid conflicting situations. In the same fashion as our personality model, we can also use different lower-level emotion traits to link to the higher-level ones with different weights to represent the subtle differences between two similar higher-level emotion traits from different models. Again, with the help of the correlations between different emotion models, we can scale our emotion model easily for different application areas, as well as for different detail requirements.

3.2.4. Action

Each action (or behaviour) contains a set of preconditions which must be true in order for it to be qualified to be executed. The precondition can vary from requirements on values of psychological states, to requirements on past social interactions, or even the current context or situation in the game. The users can configure criteria for values of personality traits or/and emotion traits they may want to use, as well as for the social tie strength. This can be achieved by setting ranges on the values mentioned as prerequisites. In addition, the past interaction history can also be used as precondition; for example, to ensure that a person may only retaliate when attacked first. A sample precondition for action: Retaliate can be described as: Social Tie Strength [-1.0,0], Joy [-1.0,0], Anger [0,1.0], Violent [0,1.0] and Was Attacked is True (see Figure 3.13). This means, roughly speaking, that retaliation happens when a character is unhappy and angry from an emotional perspective, has a tendency towards violence from a personality perspective, and was attacked recently by someone with whom they have a generally negative relationship with.

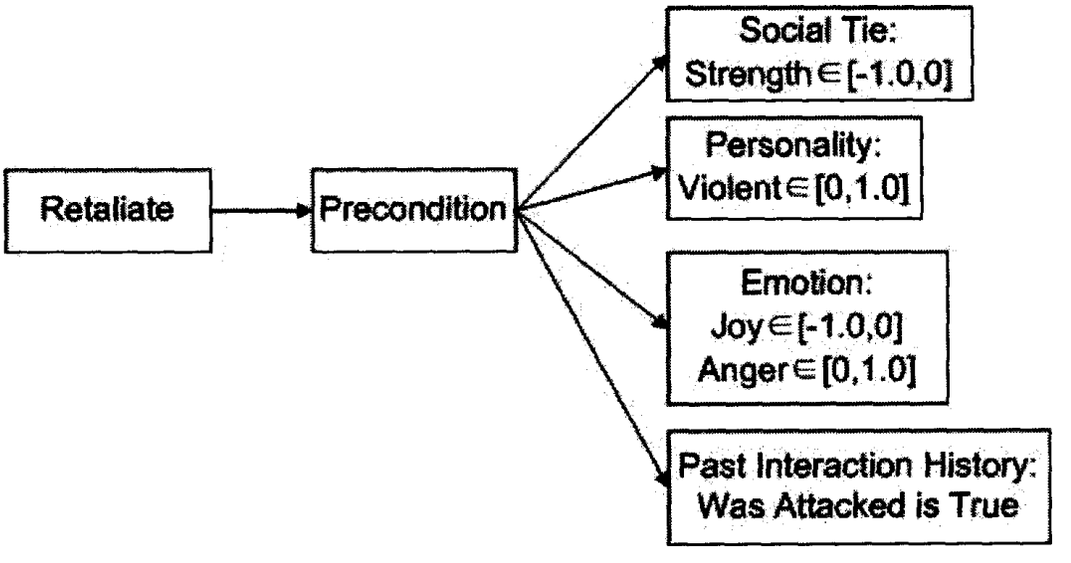

Figure 3.13: An illustration of the precondition of retaliate.

A set of postconditions (or effects) is also included in each action to specify the effects (psychosocial stimuli) caused by it. The effect can be modifications to the values of the internal states of NPCs, including emotion changes, social tie strength

changes and so on. Those modifications could be regarded as the crude responses to the psychosocial stimuli, comparing to the deliberate responses in the form of goal-oriented actions. For the effects caused by the action, the users can indicate the receivers, either action sender or action receiver, or both depending on context. The reason of allowing an action sender to receive stimuli is that most of the time, people will be emotionally affected by what they did. For instance, a policeman will feel guilty by killing an innocent person by mistake and a guilty person will feel relief after confession. A sample postcondition for action Attack can be defined as: To action sender: Joy+0.01, Anger-0.02. To action receiver: Joy-0.02, Fear+0.03. Social Tie Strength-0.025 (Figure 3.14). This means, roughly speaking, that attacking someone hurts the relationship with that individual, makes the attacker slightly happier and less angry, makes the individual attacked less happy and more afraid, and causes both the attacker and the individual attacked to remember the incident.

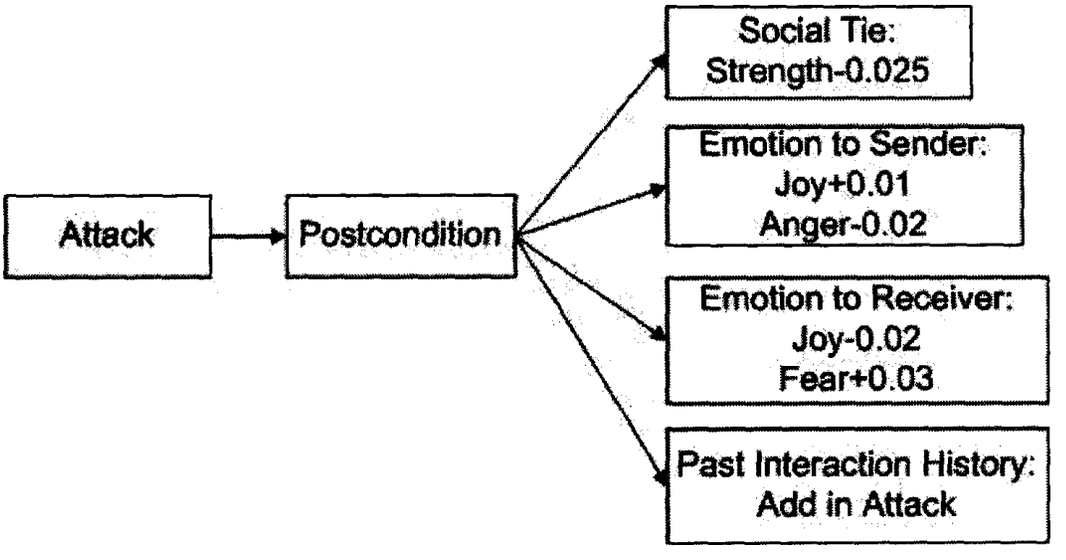

Figure 3.14: An illustration of the postcondition of attack.

The modifications to NPCs' internal states will result in an unexpected second order of consequence from the behaviours chosen for next interactions. With another layer of goal filtering mechanisms, as discussed below, emergent behaviours that were not planned, yet believable and realistic, will be presented to the players in games. In the next section, the model of goal will be discussed to construct the goal mechanism.

3.2.5. Goal

A goal is defined as a set of preferred game world states. The user can define goals as some desired internal states of a certain NPC, as well as a desired social relationship strength value between two NPCs. A sample for this is that a mother can have a goal of keeping her child happy, a husband can have a goal of keeping his wife in love and an employee can have a goal of keeping a good relationship with the boss. As samples, a goal for mother can be defined as: To Child Joy [0.5,1], a goal for husband would be: To Wife Love [0.6,1], and a goal for employee would be To Boss Social Tie Strength [0.4,1]. Figure 3.15 describes a typical set of goals for a parent as an example.

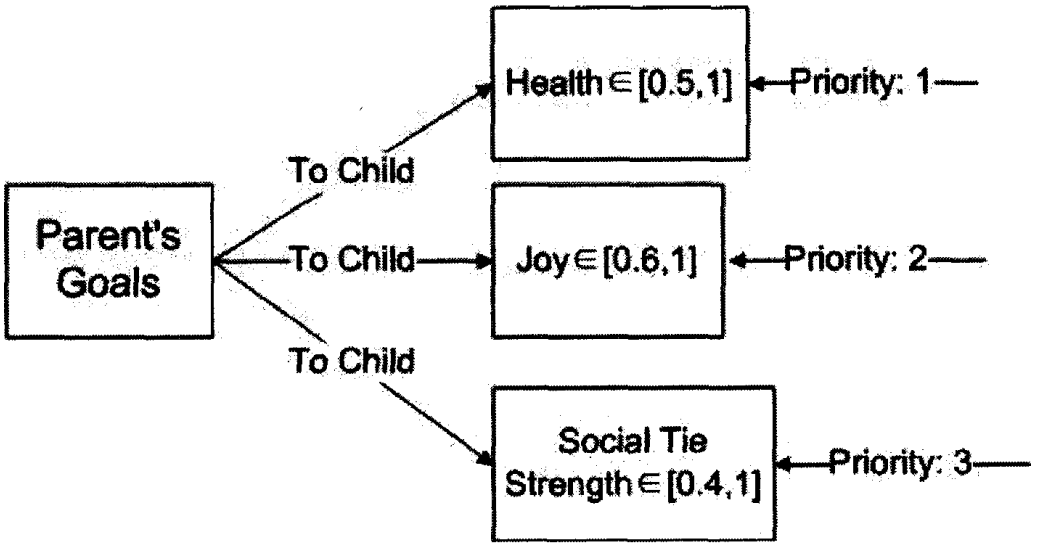

Figure 3.15: A typical set of goals for a parent.

However, without assigning an importance value to each goal, the NPCs have no preference over the plans to achieve their goals, but in a first-come, fist-serve fashion. In this case, even if the NPC has some immediate urgency, for example, being chased by a bear, the NPC will still keep on looking for food instead of running away since there is no goal prioritization. Another common case is that sometimes people have conflating goals, for example, a man may be in a dilemma where he has to balance his family and his career. In this case, he has a goal of taking good care of his family and another goal of getting promoted at his company, but he only has limited resources

and energy. In this case, he can only balance his family and career according to their relative importance to him. The priority of goals can be either static or dynamic. A static goal priority configuration can be used in some typical roles to distinguish and magnify their special characteristics, for example, the safety of child means everything to a mother. A dynamic goal priority configuration can be implemented by checking the immediate surrounding and the internal states of NPCs, for example, hunger, fatigue and so on, to update the priority for each goal.

3.2.6. Role

A role is defined as a coherent set of standard behaviours (actions) and a typical set of goals in our model, as most of people in the same role share the behaviours and have the same goals in common. (Naturally, this is a generalization; our approach supports providing special cases for when characters do not necessarily ascribe to everything in one or more of their roles.) A sample is that most parents do cleaning, cooking, and other chores at home, and they have goals of keeping their children happy and healthy. Introducing this role concept into the architecture in this way can largely facilitate the reuse of resources in simulating hundreds of thousands of social agents. Roles can be associated with age, gender, profession and other attributes. One can have multiple roles (see Figure 3.16) and switch them according to the social context, for example, time, location and the person you are interacting with. For example, Jane is Kate's younger sister and David's wife at home, but she is also Kate's boss and David's assistant at the company. Norms, guides for behaviours, can also be integrated into the role model even though some of them are widely associated with culture. For example, a Chinese role can have different norms (e.g. rules of courtesy) from an American role. People sharing the same role usually have similar behavioural norms, for example, most of white-collar workers shared the norms of dressing properly at their company, wearing suitable business attire. Another concept, value, can also be integrated into a role. A value refers to a personal moral system which describes how to pursue an ideal abstract state. For the roles associated with professions, it is likely that they will share

some typical sets of values. A doctor may prefer saving lives under the risk of getting infection rather than getting away from a plague. A vegetarian would rather suffer from hunger than eating meat. A soldier may treat discipline so important that he/she is unlikely to retreat from the battlefield without being ordered to do so. Values can be modeled using a name and a weight indicating the importance of this value in the decision-making process. Roles will be switched automatically depending on the social context, which can also be integrated into a role. Context is represented as time and place in our system, which indicates when and where the associated role will get activated, although other forms of context are also possible. For instance, the solider role will be activated whenever the soldier is on the battlefield and the worker role will be turned on in the daytime at the company. Multiple roles can also be activated at the same time in the form of having mixed goals and behaviours from those activated roles, which leads to more conflicting and unpredictable NPC behaviours. A role transition mechanism, using the concepts from FSM, can also be implemented into the role model by defining major relationship events that will lead a relationship from one state to another. For example, marriage will bring the roles of boyfriend and girlfriend into husband and wife, and the role of couple will be transited to friends (or perhaps enemies) through divorce. Other concepts from Guye-Vuilleme (Guye-Vuilleme and Thalmann, 2000), including a role hierarchy, associated roles and types can be integrated into the role model as well. For example, a policewoman role can be declared as a child of a police role and of a woman role with its own behaviours and values. A husband role inheriting from a man role can be associated with a wife role which inherits from a woman role. The concept of types helps to perceive the world and generate common sense, for instance, a scientist type is highly rated on intelligence and lowly rated on strength.

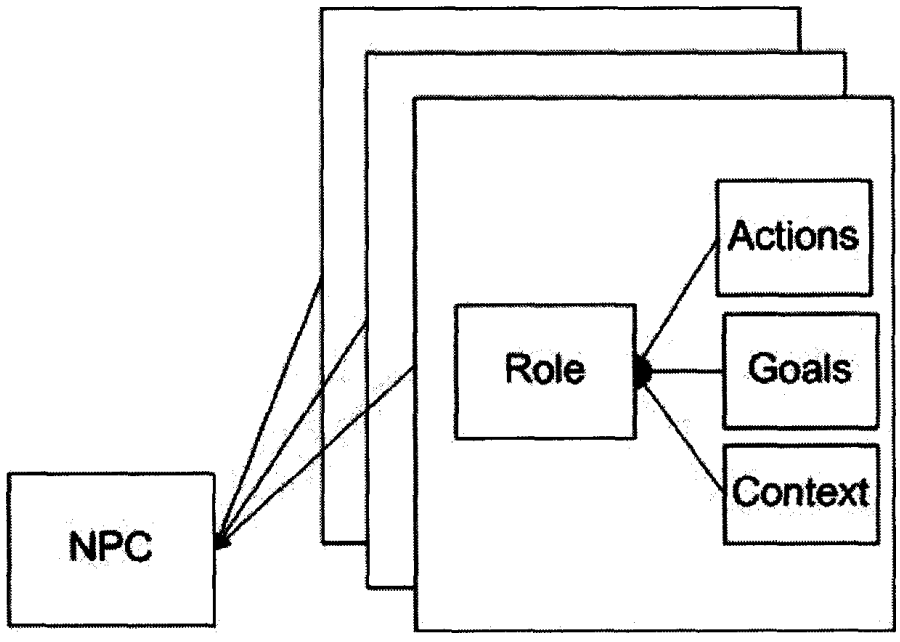

Figure 3.16: One NPC can have multiple roles.

3.2.7. Social Tie

A social tie is a concept that groups all the relations between two NPCs into a tie, which can hold multiple roles. Therefore, two NPCs can either each one have a social tie towards the other, or have no social tie at all if they have no connection in this configuration. More than one role can be assigned into a social tie depending on the relationships between these two NPCs. For example, Kate is Peter's mother and his teacher as well. For each role in the social tie, a strength value and a weight are attached on it, helping to calculate the overall strength of the social tie towards the peer. With this mechanism, we can create more realistic situations where NPCs are holding mixed and conflicting feelings towards others. (This often creates more interesting and compelling storytelling possibilities, and so supporting this is important.) An example of this is that Tom likes Jack as his younger brother with strength of 0.5 and weight of 0.6, while disliking him as his boss with strength of -0.3 and weight of 0.4. Therefore, an overall strength of 0.5*0.6-0.3*0.4=0.18 is assigned to this social tie to represent the overall positive feeling towards Jack, assuming that the social tie form Tom to Jack only involves these two roles. For each social tie the NPC has, an importance value can be given to indicate the importance of the peer NPC, and the value will be used to appraise how deep the interactions with the peer

will affect its internal states. By introducing the importance value and the appraisal mechanism, interactions with different NPCs can result into different state changes depending on the value of importance. For example, the death of an important friend will make the NPC very upset and the car accident of a stranger could cause much less depression to the NPC. NPCs can also hold memories of past interactions in social ties to help reasoning and make deliberate decisions. This memory mechanism can be implemented by maintaining a list of pointers to the actions performed between the NPC and its peer. Note that an NPC is also allowed to forget the actions that happened a long time ago or relatively unimportant events to save system memory and reduce reasoning time. We can also set up a default role for each social tie so that when there is no matching context for all the binding roles in the social tie, the default role will take effect. The overall structure of a social tie is shown below in Figure 3.17.

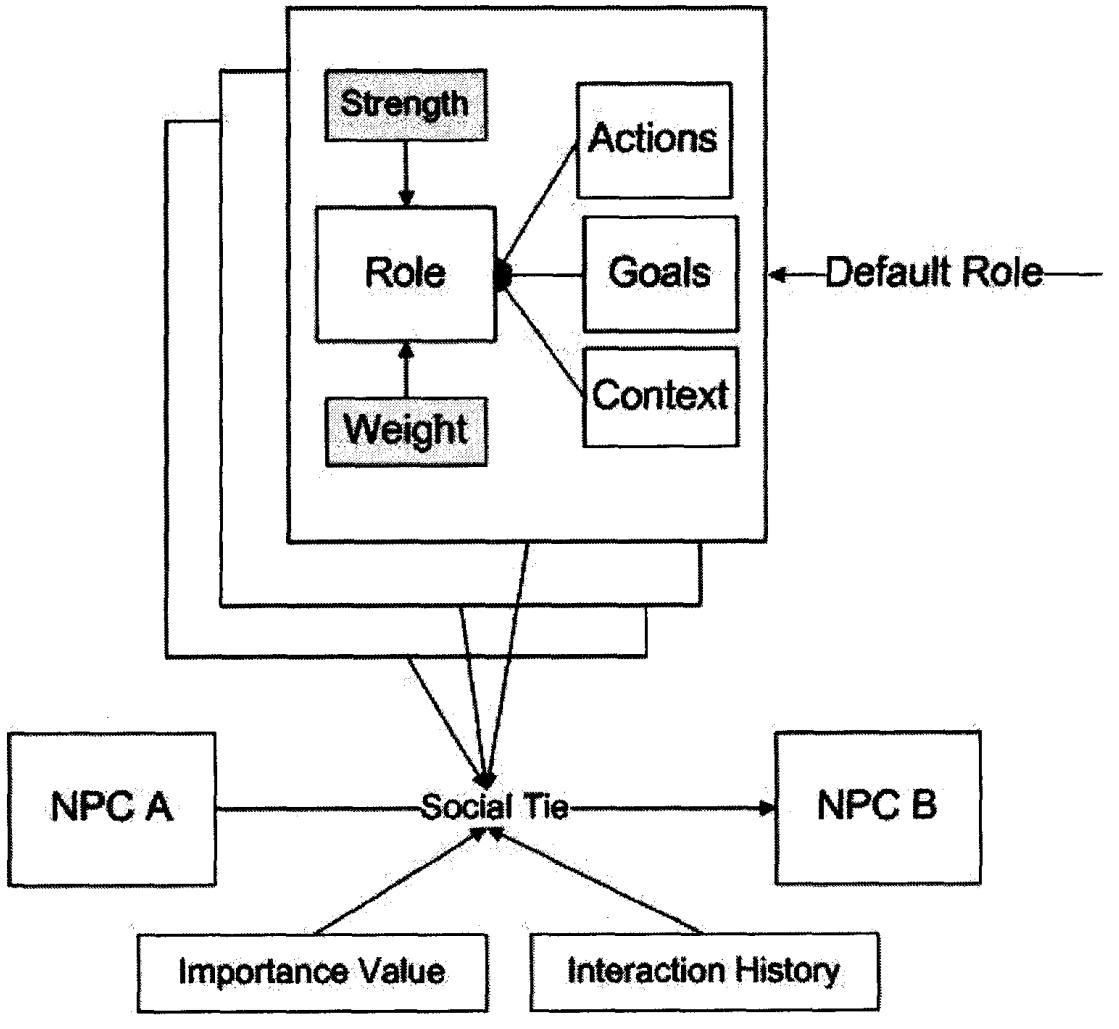

Figure 3.17: The structure of a social tie.

3.2.8. Message

The message passing mechanism in our model is designed using the Observer Design Pattern (Gamma et al., 1995), which works by notifying the NPCs who have subscribed to the message the event. Note that this is actually a publish/subscribe architecture that only those NPCs who are qualified for receiving the messages will get the message notifications from the message publisher. The event message broadcast can be an action performed by an NPC, including attributes, for example, action sender, action receiver, action id, radius of passing, effects to the social tie strengths between NPCs (between action sender and message receiver and/or between action receiver and message receiver), effects to the emotion states of the message receivers and so on. The designers can customize for each NPC for what messages it want to receive. Note that only those NPCs who are within the radius of passing and are the subscribers can receive the message. For example, a message can be described as follows:

Action: Kiss Action Sender: Kate Action Receiver: Tom, Radius of Passing: 50 units, Effects: Social Tie Strength to Action Sender -0.02, Social Tie Strength to Action Receiver -0.01, Emotion Changes to Message Receiver: Joy-0.02, Jealousy+0.03 (see Figure 3.18).

When Jack subscribed to this message, and if Jack is around within 50 units where Kate Kissed Tom, Jack will be notified of it and the Stimulus Dispatcher module will modify Jack's emotion states, as well as his social tie strengths to Kate and Tom, as Jack is in love with Kate, and does not appreciate her showing affection to someone else, particularly Tom.

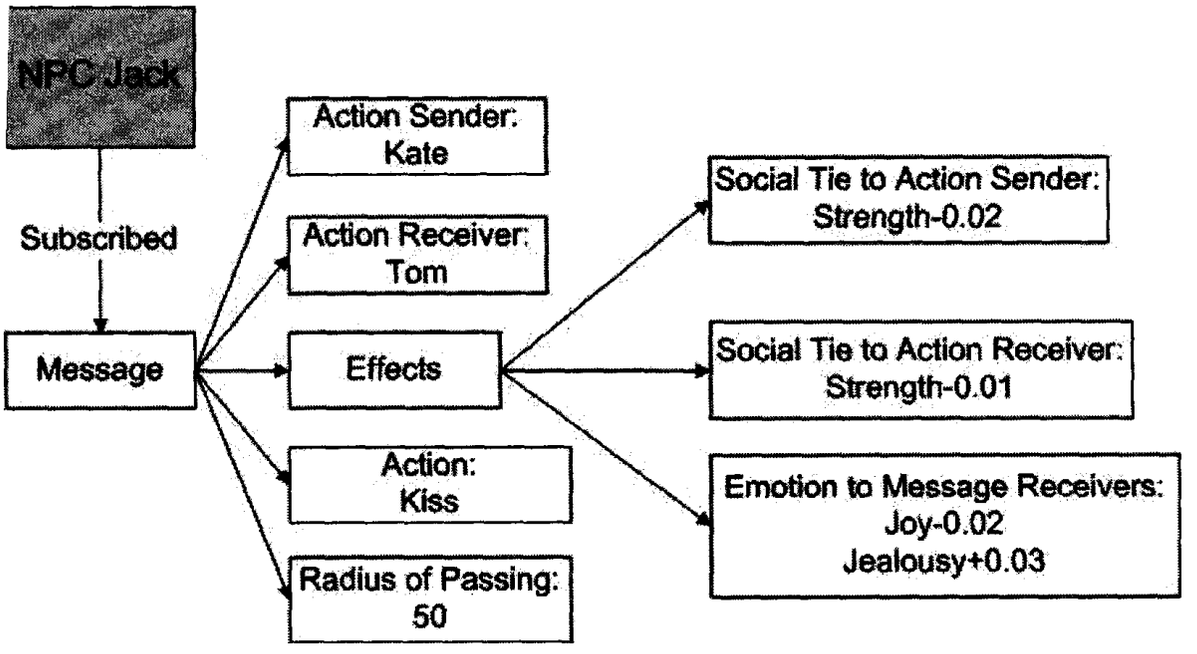

Figure 3.18: A sample of the structure of the message.

3.2.9. Data-Driven

Our model is designed to be totally data-driven, which means by changing the input data, our system can generate reasonable and flexible results without modifications to code. The flexible output of our system can be driven by the diverse data coming from the personality model used, emotion model used, action preconditions design, action postconditions design, message effects design, goals design and so on. To obtain the benefits of being data-driven, most of the important values and model designs are read from data, instead of hard-coded into the system.

3.2.10. Stress and Coping

Stress is caused by stressors, which can be actions received or events perceived. Human behaviours are directed by intrinsic stress in some sense. Physical stress and emotional stress can come from the problems faced and events occurring, more or less, depending on context. For example, the death of a spouse could cause considerable stress, while business readjustments could cause relatively less stress. Another example is that suffering from hunger will mainly cause physical stress, but divorce will mainly cause emotional stress instead. Coping can be viewed as reducing the

suffering stress caused by stressors in some sense, and it can be divided into two aspects: solving the problem (problem focus) and addressing the emotions (emotion focus). The goal for coping is to solve the problems and to address the issues at hand, and finally let ourselves stay calm and live in peace again. Consequently, coping and stress mechanisms can work closely with goals, and guides the NPCs throughout their lives. We can configure a mechanism that causes certain kinds of actions and events to increase certain kinds of stresses in various amounts depending on the context. On the other hand, those actions and events can also reduce other sorts of stresses in different amounts. For example, breaking up with a girlfriend could raise emotional stress, which could make people emotionally upset while alcohol and tobacco could somewhat alleviate this kind of emotional stress, but increase health stress at the same time. Every NPC is trying to cope with stresses and be less stressful. NPCs can have different stresses varying from physical stresses to emotional stresses and each stress is assigned with a priority value according to the current stress value or NPC's internal states. The value of stress can be evaluated from how NPC's psychological states and physiological states depart from the normal values. Usually, the higher the stress value is, the higher priority it will earn. Decreasing the values of those stresses with high priorities become goals of the NPCs. A sample is that, people suffering from divorce will bear a very stressful emotional state, sad and anxious. To decrease this emotional stress become a goal and some of them will rely on alcohol and tobacco to achieve that at the cost of increasing their physical stress. Gradually, their health stress will go beyond their emotional stress and become their priority. In this configuration, NPCs will need to use an inference module to reason how to accomplish their goals of being less stressful with the incomplete or complete knowledge of the consequence of the actions and events. The consequences can be in the form of the preconditions and postconditions, and postconditions can be the modification to various stresses in different amounts. Figure 3.19 depicts a sample of the coping with stress mechanism.

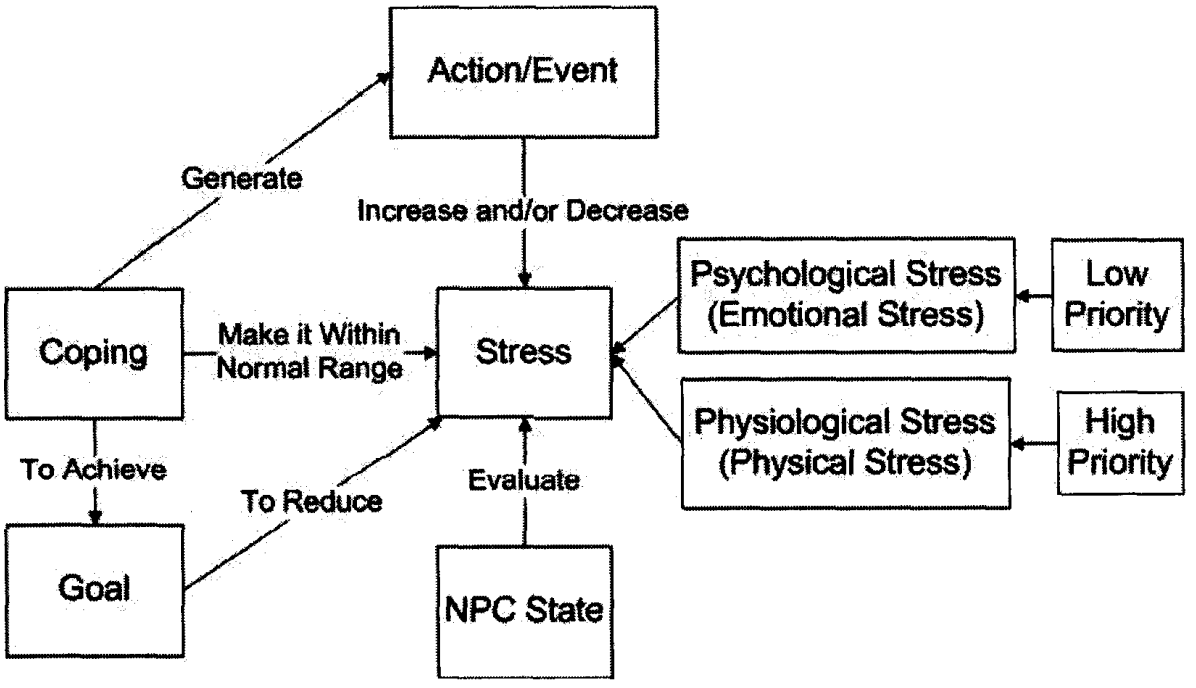

Figure 3.19: A sample of coping with stress mechanism.

3.2.11. Need

Need is a physiological or psychological condition that must be satisfied to remain healthy (Beaumont, 2005-2008c). Human needs can be classified as either physiological needs (those required to sustain and grow a healthy body) or as psychological needs (those required to sustain and grow a healthy mind). Physiological needs can include air, water, food, shelter, sanitation, sleep and touch while psychological needs may contain autonomy, competence and relatedness.

In Maslow's well-known hierarchy of needs, he divided the needs into survival need, security need, social need, needs of psychological safety, cognitive need, aesthetic need, self-fulfillment and self-actualization (Maslow, 1946). Each of these needs can be modeled using a name and a value, and goals can be defined as objectives to meet those needs. Needs, goals and stresses are tightly connected, and some primary, basic and common goals can be described as objectives to meet the needs and to reduce the suffering stresses. The hierarchy of needs can help to define the priorities of stresses as well, as the stresses related to survival are likely to have a high priority (see Figure 3.20). For example, a person may suffer from the same amount of stress from both

hunger and divorce, due to the fact that survival need is more crucial than social need, therefore, stress from hunger is given higher priority in this case. The categories of needs can help us to classify different kinds of stresses. After all, the source of stresses comes from the fact that people's needs cannot always be satisfied.

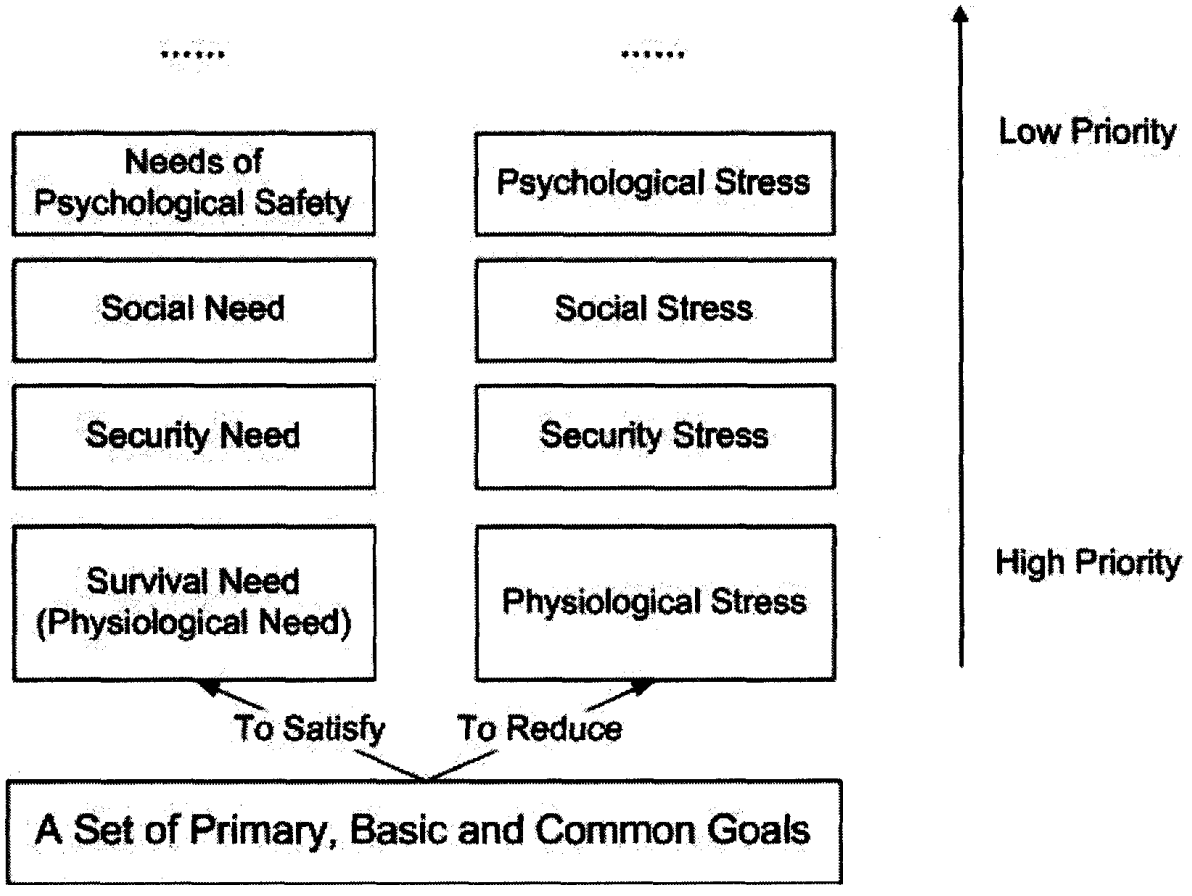

Figure 3.20: The correlation among needs, stresses and goals.

3.2.12. Life Guidance System

This Life's Guidance System (see Figure 3.21) gives us an inspiration on how to integrate all the psychosocial components into our system to guide our autonomous NPCs to act believably throughout their lives (Beaumont, 2005-2008b). This system shows us the logic behind the necessary psychosocial concepts that help us travel through our lives; therefore applying this logic into our NPC system is likely a good enhancement of believability. For example, it shows the relationship between goals and needs, as well as the correlation between stress and coping.

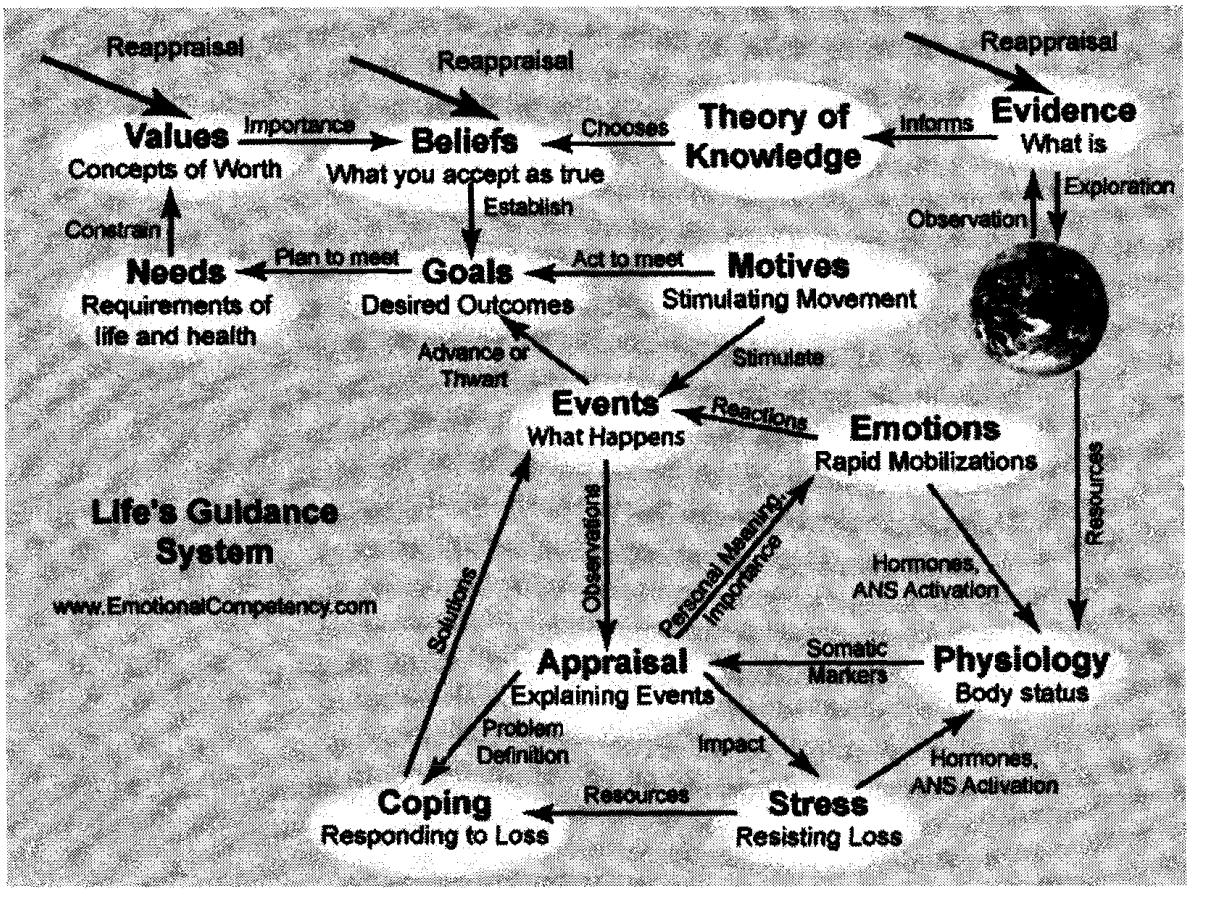

Figure 3.21: Life's Guidance System (Beaumont, 2005-2008b).

Inspired from the Life's Guidance System, we explored the underlying logic and used a similar mechanism to integrate the psychosocial components into our believable NPC architecture (see Figure 3.22). First of all, actions and events will proceed to the Relevance Filter to check if the NPCs should receive the stimuli or not. The filtered stimuli will then be passed to the Appraisal module to generate various personal effects depending on the importance value from the NPC's social ties. The Stimulus Dispatcher is then responsible for the delivery of the effects. The internal states of the NPC are used to evaluate how stressed the NPC is. The Coping module, including an Action Filter and Goal Filter, is used to generate a deliberate and goal-oriented action under the influence of the NPC's internal states to reduce the suffering from stresses and to achieve the goals. However, an action or event to advance certain goals and to reduce certain stresses may thwart other goals and increase other stresses. The Coping Module has the capability to reason about the consequence of an action and an event so as to meet the goals in order and reduce the stress with priority. Primary goals can

be defined as to meet the needs that would otherwise generate stresses, so that those basic goals can be defined as the attempt to reduce the suffering stresses as well.

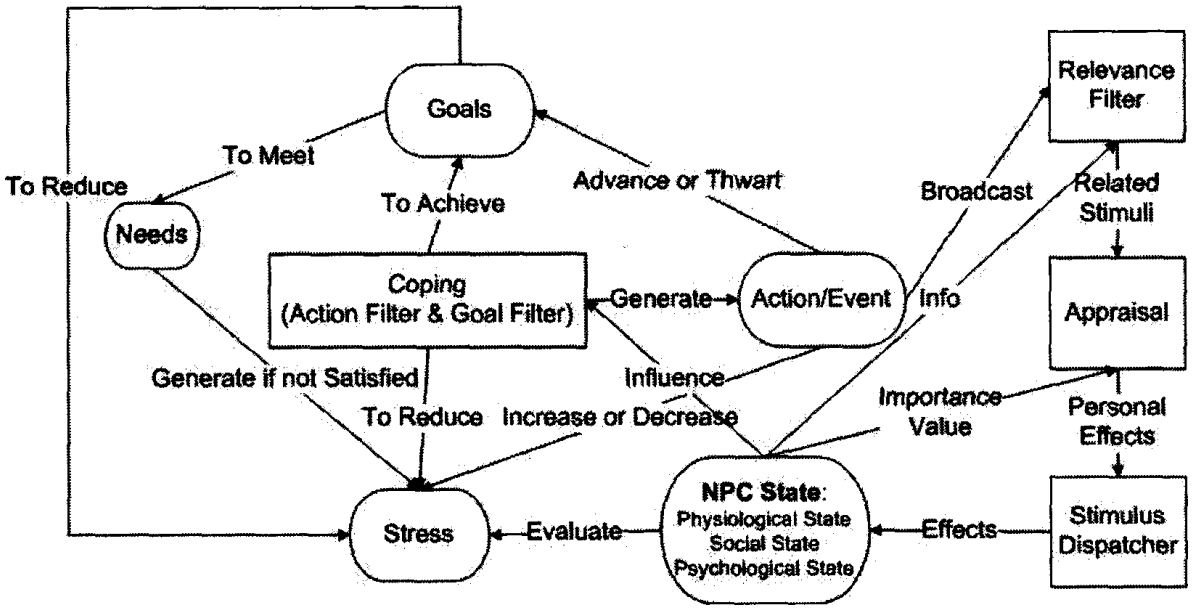

Figure 3.22: The integrated guidance system in our believable NPC architecture.

This integrated guidance system will be discussed further in Section 3.4 with respect to the various corresponding our NPC architecture.

Even though we discussed typical and stereotypical roles as examples before, we provide tools to knowledgeable experts to enable them to create the roles as well as other psychosocial models. For example, they can create a library of reusable models and they can also create a model specific to the story/game experience that they want to deliver. The next section of this thesis will examine the design of such tools.

3.3. Comprehensive Believable NPCs Creation and Management Tools Design

Our approach to believable NPCs is designed to be totally data-driven, which means the users have the freedom to setup the system configuration as appropriate and create their own NPC profiles for their games. The system configuration data includes the

definitions of personality models, emotion models, the preconditions and postconditions of actions, goals, roles, messages, and so on. The NPC profile data holds a set of personality trait values, a set of initial emotion trait values, a list of social ties configurations and a group of messages subscribed. Figure 3.23 presents an overall construction of the tools and their connection to the run-time system.

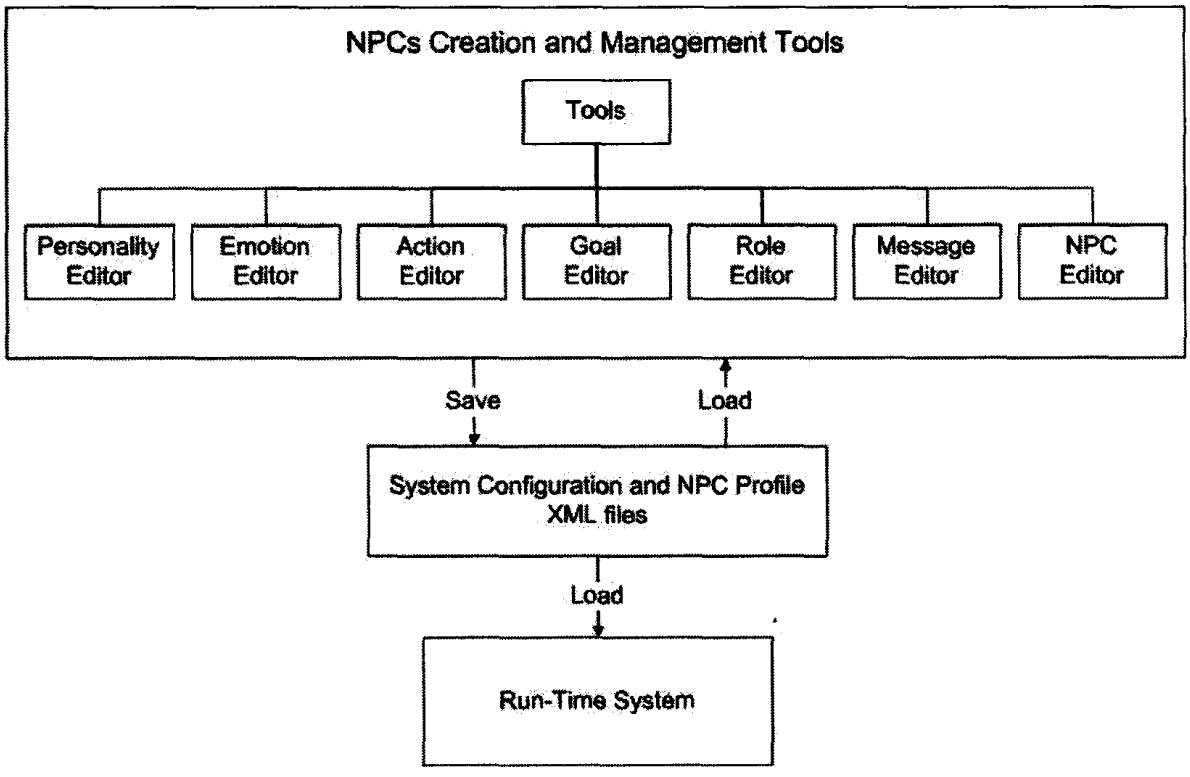

Figure 3.23: An overview of Comprehensive Believable NPCs Creation and Management Tools.

For personality model definition, the users can create different models by assigning various different traits into different places in our two-dimensional model and indicating how these traits correlate with each other in the Personality Editor.

In the Emotion Editor, the users can define different emotion models by specifying the place and correlations to others for each emotion trait in our two-dimensional emotion model.

For the definition of an action, the users can create various prerequisites and effects as preconditions and postconditions in the Action Editor.

The Goal Editor allows for goal definition, and different ideal game world states can be specified as goals for different NPCs to pursue.

For role definition, the users are able to create different kinds of roles by using different standard actions and different goals already defined in the system.

For message definition, the users can create different messages by specifying different events or actions, the various involved people, and different stimuli attached.

For NPC definition, different NPCs can be created by assigning different personality trait values and different initial emotion trait values to different individuals, as well as creating different social ties and different subscribed messages for different NPCs.

The system is driven by the definitions of those elements and the values embed in each NPC; therefore different definitions of the components and different values embedded in the NPCs can result in completely different system output. All data designed and composed by the users is stored in the form of XML files (configuration XML files and individual NPC profile XML files) for the run-time system to process and construct the game world from. The system is neither hard-coded nor scripted; instead it is completely driven by the NPC data and the rules defined in the input XML files.

Since the system's proper functioning relies heavily on the quality of the input data in the XML files, we developed tools to help the designers to design and maintain their configuration and NPC profile data with minimal effort. These tools are based on a graphical user interface (GUI), and they are capable of loading and saving the generated data in the XML file format, while being easy to use and general enough to create and modify configuration and NPC profile data. Implementation details of these tools are discussed further in the next chapter.

3.4. Emergent Framework Design

When designing the architecture of believable NPCs, we need to find out what factors will influence and motivate believable NPCs behaviours in a social context. Without considering the basic physiological needs (such as needs for food, water, sleep, and safety), the factor that most drive our behaviours are goals. NPCs are trying to achieve their goals using different methods under the influence of their psychosocial situations.

The emergent framework design used in this thesis is based on elements from Bailey's framework (Bailey, 2007) to reduce development efforts, with modifications and extensions to support the needs of our current work. Relevant aspects of Bailey's framework will be presented in this section alongside our own framework, to have a complete discussion of the system in place. Elements from Bailey's framework will be clearly cited and colour-coded to avoid confusion.

The following sections outline the mechanics of an emergent NPC social psychology (3.4.1), mapping stimuli to actions in an NPC psychology (3.4.2), message passing mechanism (3.4.3) and the architecture of a believable psychosocial NPC (3.4.4).

3.4.1. The Mechanics of an Emergent NPC Social Psychology

In order to use emergent techniques to build our believable NPCs, we have to understand the mechanism of a typical emergent system. An emergent system uses interactions of component-level behaviours to form system-level complex behaviour. Those component-level behaviours can be quick responses to stimuli, and every NPC in the game world is responsible for its own sensing and responses to those stimuli.

3.4.1.1. An Abstract Stimulus-Response System

A stimulus-response system operates by having objects create and respond to stimuli. The objects that create the stimuli are referred as the interaction Actors. On the other hand, objects that receive and respond to the stimuli are called interaction Reactors, which could become Actors through their responses. Each Reactor might react to different stimuli and react to the same stimulus differently. The stimulus messages can hold attributes like the effects of the stimulus, radius of the stimulus, the type of propagation and so on. By applying some modifications to this abstract Stimulus-Response system, a psychosocial emergent system can be created.

3.4.1.2. A Stimulus-Response System for Psychosocial Behaviours

In a psychosocial context, those Actors and Reactors could be NPCs and events, and the stimuli could be actions and messages of events. The attributes associated with the stimuli can include the psychosocial effects to the receivers, the radius of effect, stimulus type and so on. The stimulus messages can also hold the reference to the Actor object that created the psychosocial stimuli, which is useful for the Reactors to react directly to the Actor. NPCs in a psychosocial context will in general listen to all types of psychosocial stimuli; however, they are likely to ignore irrelevant events and only react to the stimuli that they are interested in. Therefore, NPCs only listen for psychosocial stimuli that they are concerned about using a message publish/subscribe architecture, as discussed earlier. Note that an NPC can be the Actor and Reactor of the stimulus it created at the same time. For example, an NPC might feel so angry that it decides to attack another NPC, which causes the stimulus of being less angry to itself and the stimulus of becoming more afraid to the attacked NPC. Figure 3.24

shows an abstract psychosocial stimulus-response system diagram modified from Bailey's emergent framework.

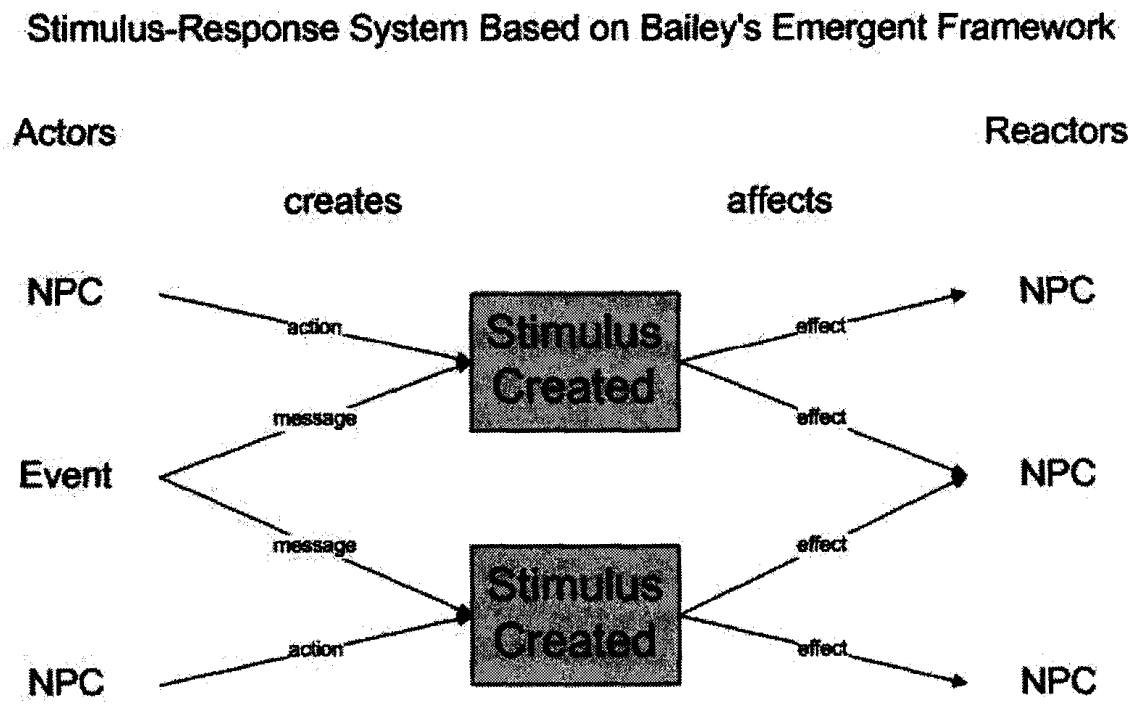

Figure 3.24: An abstract stimulus-response system for psychosocial behaviours.

If we design the psychosocial mechanism as the way presented in this section, NPCs can react to psychosocial stimuli in flexible as well as dynamic and unexpected ways to their environment. Chain reactions as well as a second order of consequence will be generated automatically as reactions to the psychosocial stimuli are likely to cause more psychosocial stimuli that can be reacted to. This would achieve the goal of having emergent psychosocial behaviour in NPCs. Additionally, with players' participation in this mechanism through their actions in the game world, emergent gameplay will be produced.

3.4.2. Mapping Stimuli to Actions in an NPC Psychology

We can define behaviours by social roles, so that behaviours are linked to these roles

instead of specific NPCs, which can facilitate the reuse of the behaviours. The next step is to associate the behaviours with internal states (emotion traits, personality traits, social ties strength and so on) that influence the NPC's decision-making processes. For example, the trait of extraversion will be positively correlated with behaviours involving seeking or interacting with other people and the trait of anger will be positively correlated with violent behaviours and negatively correlated with rational behaviours. Similarly, NPCs with high introversion will be less likely to select those outgoing behaviours. Finally, stimuli and NPC internal states are correlated by the effects that each stimulus is carrying, which will act on an NPC's internal states eventually. Figure 3.25 describes the whole process to us.

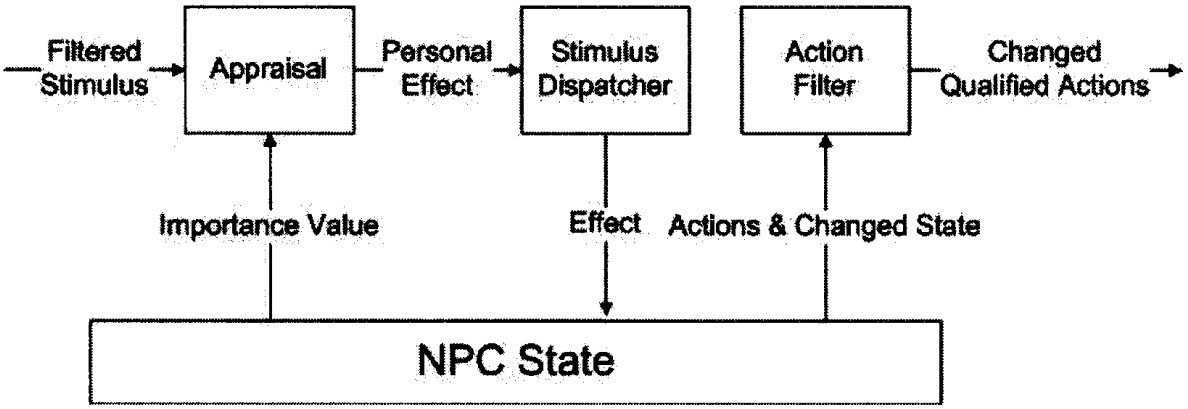

Figure 3.25: The processes of mapping stimuli to actions.

3.4.3. Message Passing Mechanism

By defining a set of behaviours or events that can be reacted to, a set of stimuli that the events create, and a set of social objects that can react to the stimuli, the mechanism explained in the previous sections can be implemented.

For the NPCs to perceive the game world, instead of actively probing the environment, the NPCs can simply listen for/receive the messages subscribed by broadcasting. When an event occurs, the Actor actively publishes the event message to all of its event listeners/subscribers. With the help of this publish/subscribe architecture, the NPCs or events in the emergent psychosocial system can actively notify the other

NPCs the stimuli they are causing.

3.4.4. Architecture of a Believable Psychosocial NPC

The architecture of a believable psychosocial NPC involves the processes of handling the input of a stimulus as well as outputting a goal-directed behaviour. Some stimuli may be filtered by relevance and thus not cause any effect and resulting output behaviour.

3.4.4.1. Handling a Stimulus

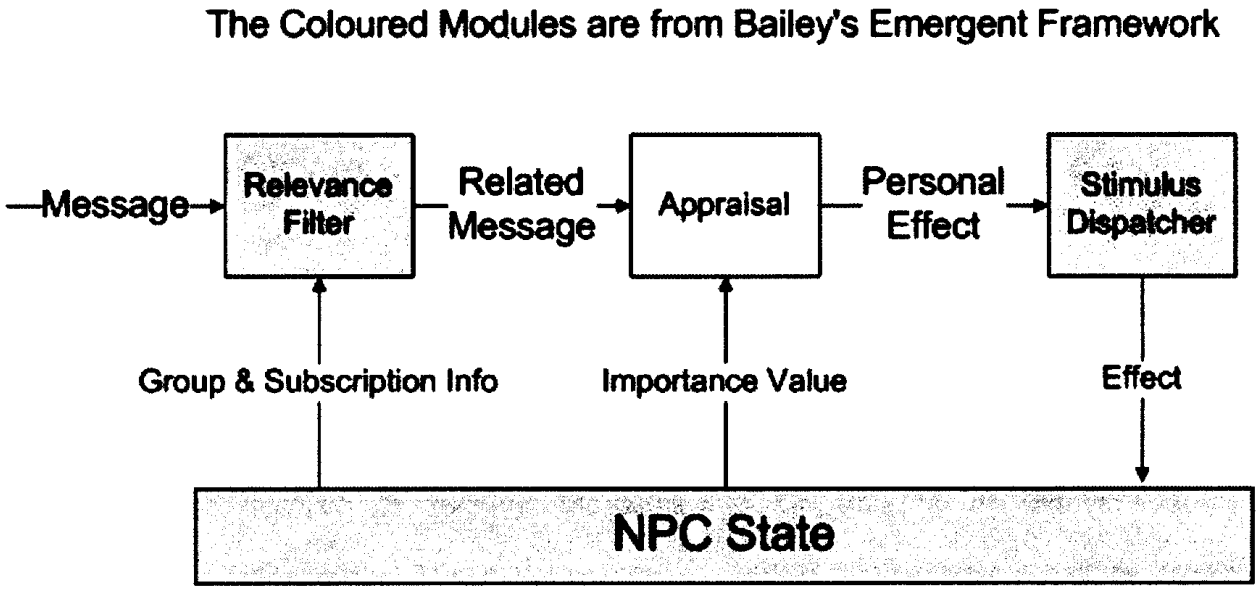

Figure 3.26: The processes of handling the input of a stimulus

In order to handle the event message appropriately, the message must pass through a Relevance Filter, an Appraisal module and a Stimulus Dispatcher. The NPC will provide access to its state to these processes as needed (see Figure 3.26). The details of the processes will be explained in the following sections. The Relevance Filter, Stimulus Dispatcher and NPC State modules are ideas from Bailey's framework and they will also be discussed to present the complete structure of the emergent system as a whole.

3.4.4.1.1. Relevance Filter

The Relevance Filter uses the message attributes to decide whether or not the message is related to the NPC. The Relevance Filter can have several stages to process the message at hand. As an example, a typical set of filters can include a group membership filter, a subscription filter, propagation method filter, a radius filter and a magnitude calculation of the message affect (see Figure 3.27). The subscription filter and radius filter are additional filters to Bailey's framework for the needs of our data models.

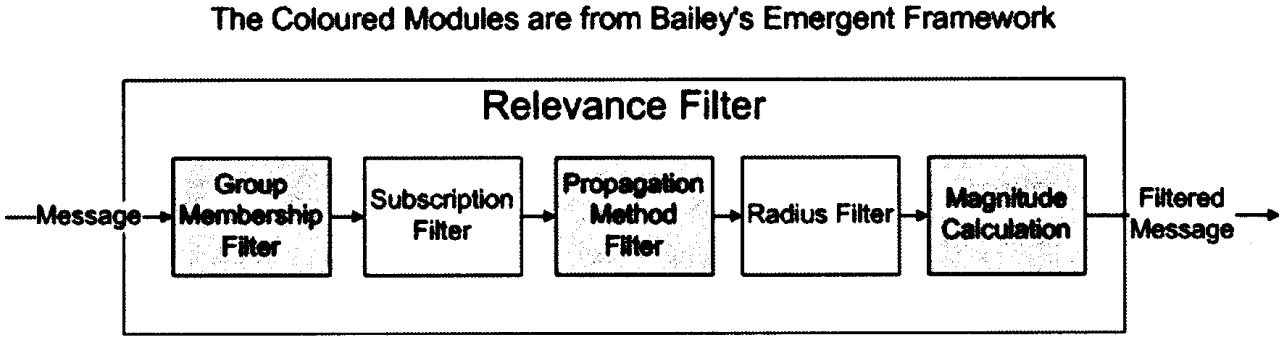

Figure 3.27: The structure of the relevance filter.

The group membership filter compares to see if the NPC is one of the members of the groups that deserve to receive the message by looking at the NPC's group membership information. If not, the message is ignored; otherwise it will be processed by the next stage.

The subscription filter simply takes at look at the NPC's subscribed messages list to see if the NPC is one of the subscribers to this event message. If yes, the NPC is eligible to proceed to next stage.

The propagation method filter simply checks if the NPC can sense the message or not. For example, if the message propagation method is by sight (the NPC can only sense this message if it is looking at the direction of the source of the event message without

blocking), it will determine if the NPC is looking at the right direction without an obstacle blocking it. Instead, if the propagation method is by phone, the filter will check if the NPC is on the phone with the sender.

The radius filter checks if the NPC is within the distance of this event message, if not, the message will be ignored. For example, if the message propagation method is by sound and within 10 meters, the filter will check if the NPC is close enough to the source of the event.

The magnitude calculation of the stimulus affect is only needed if the propagation method is by radius and has a fall off. It will calculate how much of the stimulus effect remains for the NPC to sense.

Once the Relevance Filter figures out that the message is in fact relevant to the NPC and what its remaining effect is, the related message is passed on to the Appraisal module.

3.4.4.1.2. Appraisal

The Appraisal module is used to examine the events perceived or actions received and generate personal meanings to the NPC. Note that each NPC will be affected by the same stimulus in various amounts depending on how the NPC appraises the event or action. For example, being betrayed by a stranger is less harsh than being betrayed by a best friend. The key to implementing the appraisal mechanism is the evaluation of how important the event or action is to the NPC. For example, if an event or interaction involves the people in the NPC's most important social ties, it is likely that the NPC will have a larger impact from the stimuli. Therefore, in our framework the Appraisal module will read in the importance values from the NPC's social ties to appraise the events and generate personal effects.

When appraisal is finished, the generated personal effect will be handled by the Stimulus Dispatcher.

3.4.4.1.3. Stimulus Dispatcher

The Stimulus Dispatcher is responsible for delivering the psychosocial effects that attached in the stimuli of actions and event messages to the appropriate NPCs. The designers would set the information about the effects to the NPCs in each stimulus in data. The psychosocial modifications made to the NPCs can be regarded as some sort of reactive, primary and crude responses to the stimuli received. On the other hand, the believable NPCs are also able to make deliberate goal-oriented responses to the environment.

3.4.4.2. Outputting a Goal-Oriented Behaviour

In this section, the processes of generating a deliberate goal-oriented behaviour will be discussed. Note that, in our design in this architecture, only if the NPC has unfulfilled goals will it proceed to output the deliberate behaviour. This can be implemented by checking if the NPC already achieved all its goals before initiating the processes shown in Figure 3.28.

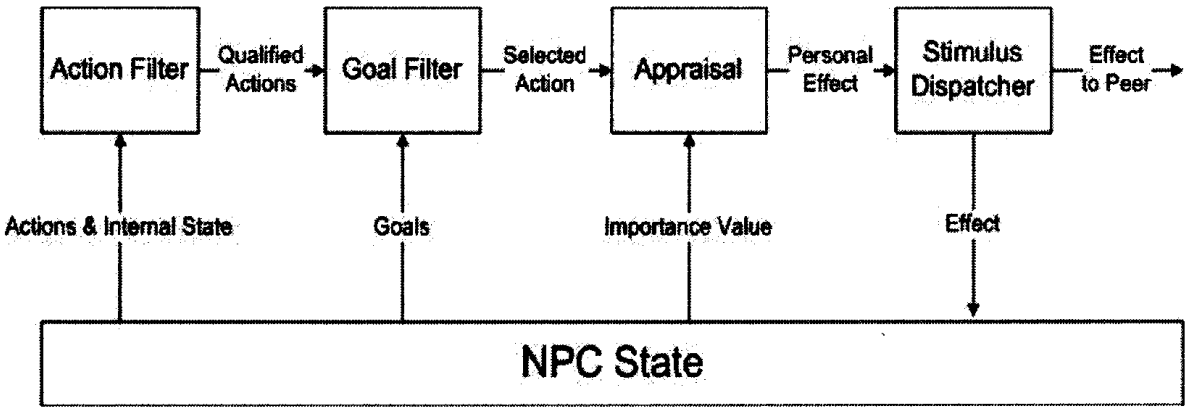

Figure 3.28: The process of outputting a goal-oriented behaviour.

3.4.4.2.1. Action Filter

The Action Filter consults the internal state of the NPC, as well as the available actions attached in the roles that the NPC is engaged in when interacting with the peer in the social tie. First of all, the Action Filter checks each available action to see if the action coincides with NPC's internal state. For example, for an NPC who is very shy, submissive, and overcome with fear to interact with another hostile NPC, actions such as provocation, attack, greet, and curse would not match the NPC's internal state, and would not be available for use. Instead, behaviours like silent treatment, flee, reconcile are suitable for the NPC in this situation. This kind of mapping from internal states to the appropriate actions can be achieved through the use of the action precondition mechanism discussed earlier, which associates a list of psychosocial values and interaction history requirements with an action as prerequisites that need to be met in order to perform the action. Those actions that can meet the preconditions are then passed to Goal Filter as qualified actions for further selection. As mentioned earlier, designers have the ability to design the preconditions for each action in the game as they want and store them as data rather than having them hard-coded in the system.

3.4.4.2.2. Goal Filter

The goals that the NPC has and the set of qualified actions from the Action Filter are the inputs for Goal Filter. Note that each goal has a priority indicating the importance of the goal according to the NPC's current state, as discussed earlier. The Goal Filter checks each qualified action to evaluate how well the action satisfies the goals in a hierarchical order, and then gives each action a rank. The action with the highest rank becomes the selected action. For example, the goals of the car salesman NPC Tom are

to please his client, improve the relationship with his client, reassure his client, and make his client feel trustful and reliable about him in order. The qualified actions are selling promotion, giving discount, proving longer hours of customer service, providing extended warranty and sending extra timely warmhearted reminders and special offers about car insurance. All of the qualified actions can please the customers in some sense. However, only the last action can best satisfy all the goals in priority, therefore, the Goal Filter will rank this action the first place and output it as the selected action. Note that for creating various and unpredictable NPCs, we can have them make poor choices sometimes by not selecting the highest rank action. To check how well each action can satisfy the goals in order, we can simply use the action postcondition mechanism as described earlier, which assigns a series of psychosocial effects onto the source and/or target NPCs as modifications to their internal states that help to determine how well this action can realize the goals in order. Note that goals are defined as some game world states in general, which in our particular system we simplify them as some NPC psychosocial states, like feeling love, getting along very well with someone, and so on. The selected action is then sent to the Appraisal module to generate personal psychosocial effects, which will be delivered to the Stimulus Dispatcher module to be dispatched to the specified NPCs. Again, as noted earlier, the designers have the ability to design the postconditions for each action in the game as desired and can store them as data rather than have them hard-coded in the system.

3.4.4.2.3. NPC State

The NPC State holds a set of attributes that describe the NPC as a psychosocial being (see Section 3.2 for details), which can be described from internal states (psychological state and physiological state), social information (social ties and group membership) and social events that the NPC is concerned about. See Figure 3.29 for a sample class diagram of a believable psychosocial NPC State

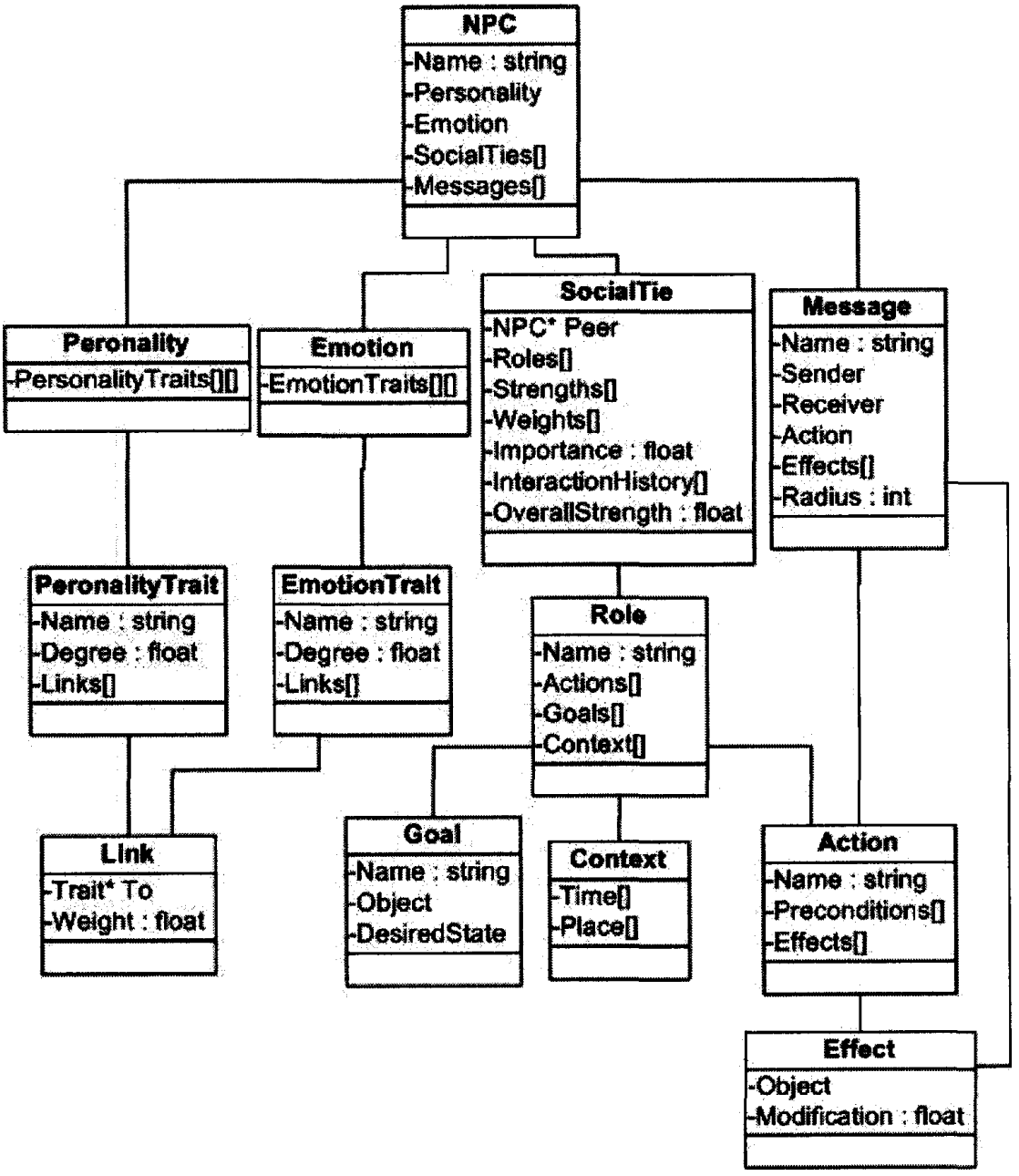

Figure 3.29: A sample class diagram of a believable psychosocial NPC State.

Both personality and emotion can be represent by a two-dimensional model (discussed earlier) filled with various traits, which are a data structure consisting of an identifying name (i.e. "shy", "joy"), a degree and a set of links to other traits. Linka are composed of a pointer to other traits and a weight indicating the relationship between them (as discussed before). To enable the flexibility for game developers to change the trait definitions for their own systems, trait information are read in from data.

Social ties can be either symmetric (two NPCs feel the same way about each other) or asymmetric (allow different views on the relationship). Each social tie has a pointer to the peer involved in the relationship, and an overall strength value of the relation (a high negative number indicating a very hostile relationship for example, a high positive number indicating a very friendly relationship, with 0 indicating a neutral relationship). As discussed earlier, the social tie also groups the social roles held by the NPC in the relationship. For example, Tom is Jack's brother at home and his boss at work. In this scenario, Tom's social tie to Jack will be labeled with roles brother, as well as boss, switching on the context. Each role is associated with a strength value as well as a weight to simulate some complicated social relationship. For instance, Tom likes Jake very much as brother, but dislikes him as his boss. Given this situation, Tom has a positive strength value towards Jack in his brother role, but a negative strength value in his boss role, therefore an overall strength value for the relationship can be calculated using the strength and weight assigned for each role. The importance value is used to evaluate how deep the interaction in this social tie will affect the NPC's internal state as an appraisal mechanism. A diagram of our social tie model is shown in Figure 3.30.

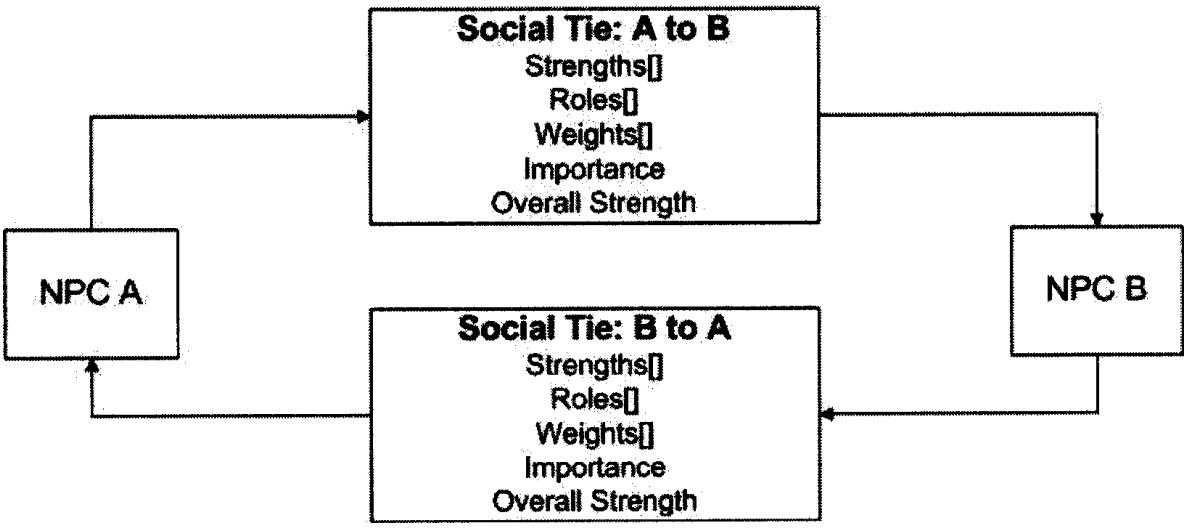

Figure 3.30: A diagram of the social tie model.

Group memberships can be represented as a list of groups that the NPC belongs to.

The role holds a set of typical actions, a group of goals and the context for role