Utrecht University

HANDLING CONFLICTS IN A MULTI-AGENT SYSTEM FOR E-COACHING

BY

SUMAIYA SULTANA RIKA
STUDENT NUMBER: 5889960

SUPERVISED BY

DR. IR ROBBERT JAN BEUN
PROF. DR J-J.CH. MEYER

August 15, 2018

Acknowledgements

I would like to use this opportunity to thank my daily supervisor, dr. ir. R.J. Beun for his support and guidance. Without his guidance, this thesis would not have been possible for me to accomplish.

I would also like to thank my second supervisor prof. dr. J-J. Ch. Meyer letting me do the thesis under his supervisions. I have learned a lot from my supervisors and I will be always grateful to them.

I would like to thank my parents, parents in law, and my husband for supporting and loving me unconditionally. Lastly, I thank my friends and well-wishers for always wishing me best.

Abstract

According to the World Health Organization (WHO), in 2016, there were over 250,000 mobile health applications in major app stores [1]. However, most of the apps are only focused on one disorder, and there are very few apps (among the listed apps) that support multiple disorders simultaneously. Whereas in reality, often multiple disorders co-occur as one disorder leads to another disorder. For instance, diabetes is one of the substantial risk factors for heart diseases [2]; sleep disorders (insomnia and hypersomnia) co-occur with depression [3, 4]; substance abuse is associated with the depressive mood, anxiety disorder, and anti-social personality disorder [5]. We model a basic framework (the so-called MentalHeath-Care or MHC) of a e-coaching system where a coach coaches his/her clients via electronic media, i.e., internet to support numerous health problems simultaneously and independently. The unique part of the system is that it would be a fully automated e-coaching system where there will be no human coach to coach the clients, but autonomous intelligent agents.

We start the framework for two relatively close disorders, i.e., insomnia and depression which often co-occur. For two distinct disorders, we plan to have two autonomous agents (coaches) who will be situated in the same setting (smartphone) and will share the same environment. Thus, MHS is built on the multi-agent system paradigm. The e-coaching system would be able to offer three types of coaching: 1) independent depression coaching, 2) independent insomnia coaching, and 3) simultaneous depression and insomnia coaching based on a client's symptoms. However, to ensure the third kind of coaching the coaches require to work in a team and cooperate with each other as either of them might have incomplete or no knowledge of the other disorder. In order to have teamwork and cooperation, the coaches need to interact and communicate with each other. But, an important shortcoming of cooperation in a multi-agent system is that it may generate conflicts or disagreements between coaches on various aspects. The conflicts may regard to different knowledge, goals [6], plans, desires, and beliefs [7] of the coaches; shared resources and task interdependencies [8] between the coaches. In this thesis, we propose a conflict resolution mechanism for a multi-agent system based fully automated e-coaching system that offers coaching on different mental disorders, i.e., depression and insomnia.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

According to World Health Organization (WHO), in 2016, the total number of mobile health (m-health)[1] applications (apps) was over 250,000 in major app stores [1]. These apps support diverse types of health problems, for instance, general health (i.e., fitness), chronic conditions (i.e., diabetes, heart failure, cancer (i.e., general, breast cancer), addiction (i.e., smoking, alcohol), and mental disorders (i.e., depression, anxiety, schizophrenia, bipolar). Most of the apps offer education to raise awareness while some of them support the disease detection, disease management, communication with responsible people, etc. Some of the mental health apps also provide tools in support of therapeutic techniques like behavior change. However, most of the apps are only focused on one disorder, and there are very few apps (among the listed apps) that support multiple disorders simultaneously. Whereas in reality, often multiple disorders co-occur as one disorder leads to another disorder. For instance, diabetes is one of the substantial risk factors for heart diseases [2]; sleep disorders (insomnia and hypersomnia) co-occur with depression [3, 4]; substance abuse is associated with the depressive mood, anxiety disorder, anti-social personality disorder, and affective disorders (mood disorders) [5]. Moreover, people suffering from physical health problems, i.e., diabetes also suffer from mental disorders, i.e., depression [11].

This thesis proposes a basic framework (the so-called MentalHeathCare or MHC) to support numerous health problems simultaneously and independently. We start the framework for two relatively close disorders insomnia and depression which often co-occur. The system can support either of the disorders or both disorders simultaneously based on the patient's (coachee's) conditions. We choose coaching as the delivery method of treatment over teaching (education) because it involves more collaborative approach and it adapts to the changes in a coachee's social and personal lives in contrast to teaching [12].

Typically, in e-coaching, a human coach offers coaching on attaining goals or

[1]WHO defines "mobile health" or "m-health" as medical and public health practice by mobile devices, i.e., mobile phone, patient monitoring devices, personal digital assistants (PDAs) and other wireless devices

solving problems via some media i.e., tv, radio, Internet (email), phone, etc. The human coach either records all types of coaching materials and pieces of advice in a video, audio or text formats and the client guides him-herself according to the records in the way of achieving his or her specific goals (self-guided); or the coach guides the patient via mail, message or call when the patient asks for help to the coach (guided). However, with the advancements of Information Technology (IT) and Artificial Intelligence (AI), we can model an advanced guided coaching approach where the client and the coach will have constant connection so that the client does not need to ask for help always, but the coach can also proactively help or guide the client whenever s/he commits mistakes or loses confidence in the way of achieving a specific goal. We can do this by replacing the human coach by an autonomous agent who has a significant amount of knowledge about the domain, accommodates with his[2] dynamic environment by observing and gathering information through sensors and other tools, makes independent decisions based on the information and domain knowledge, and acts upon them at the right time. In other words, we can replace the human coach with an AI agent who can mimic the behavior of the human coach.

For two distinct disorders, we can have two autonomous agents (e-coaches or coaches) who will be situated in the same setting (smart phone) and will share the same environment (the coachee and the coachee's surroundings). Our intention is to ensure three types of fully-automated e-coaching through this system: 1) independent depression coaching, 2) independent insomnia coaching, and 3) simultaneous depression and insomnia coaching based on a coachee's symptoms. However, to ensure the third kind of coaching the coaches require to work in a team and cooperate with each other as either of them has incomplete knowledge about the two disorders. In order to have team work and cooperation the coaches need to interact and communicate with each other as well.

The Multi-agent system (MAS) seems a suitable paradigm to model MHC since a MAS is defined as a group of homogeneous or heterogeneous agents who communicate and interact with each other to achieve individual or collective goals. Cooperation is a key concept of MAS [13, 14, 15, 16] which is used to establish teamwork in order to achieve collective goals or help each other in their individual goals. Cooperation in MAS also enhances the quality and quantity of tasks done by the agents as it i) improves the rate of task completion through the parallel processing; ii) increases the number of parallel tasks by sharing resources (i.e., environmental information, expertise or domain knowledge); iii) increases the chance of tasks completion by replicating previously completed similar tasks, and comprehending tasks from different perspectives; iv) decreases overlapping or interference between tasks by evading negative interactions [17].

However, an important shortcoming of cooperation in MAS is that it may

[2]For clarity in writing, from now on we will address the coaching parties with different genders, i.e., the coaches will be addressed as male and coachees as female.

generate conflicts between agents on various aspects. A conflict can be defined as a situation of disagreement between two or more agents or two or more groups of agents [7]. This disagreement can regard to different goals [6], plans, desires, and beliefs [7] of the agents; different norms or regulations [18] of the system; shared resources and task interdependencies [8] between the agents. This paper aims to model a fully automated e-coaching system using MAS paradigm and present a conflict resolution (CR) technique to identify and resolve a particular type of conflicts between the coaches of the e-coaching system.

In a MAS, conflicts occur in many cases due to various reasons and conflicts are resolved based on their types and dimensions [7]. Generally, conflict resolution involves conflict detection, search for solutions, reaching an agreement with regard to the solution to be pursued [6], and perform the solution. CR may occur at different levels of the system execution, i.e., explicit conflicts between different domains can be identified at the design phase of the system; implicit conflicts between the domains can be identified at the runtime; conflicts can also occur due to the characteristics of the application domain so conflict detection process should take into account the application domain also.

The e-coaching system built on MAS paradigm will also be inclined to conflicts largely because the system will consist of autonomous agents who may have distinct knowledge bases, sensors, and inference mechanism to make decisions, react to the dynamic environment and work on the environment proactively. There will be no centralized management scheme to control the coaches, and no coach will have complete knowledge of all the domains. Therefore, the coaches will have to cooperate with each other to provide coaching on multiple disorders.

Since the application is about health care, the current (before the resolution) and future (after the resolution) conditions of the coachee must be taken into account. Depression is a mood disorder, and its coaching may depend on the coachee's mood. Insomnia can also affect the coachee's mood. Hence, to provide coaching on both disorders, the system must adapt to its most dynamic and complex environmental component which is the coachee's mood. A person's mood can change anytime. And depending on her mood new conflicts may arise, or solutions to the detected conflicts may change. Hence, one of the main challenges of the system will be adapting to its ever-changing environment and detect and resolve conflicts based on the current situation and future result. The problem to be dealt with in this thesis is as follows.

Problem Statement:

How to resolve conflicts in a multi-agent system for fully automated e-coaching system?

To find the solution to this problem, we first need to resolve some other sub-problems, i.e., 1 What is e-coaching and how can the process of e-coaching be modeled?

2 What is the intervention technique for e-coaching in the domains of insomnia and depression? And how will the e-coaching be modeled?

3 How can a fully automated e-coaching system be realized in the MAS paradigm? And how can the system accommodate to the dynamic environment?

4 What kind of conflicts may arise in the e-coaching system between the coaches?

5 How can the conflicts be detected and resolved at different levels and times?

The chapters in this thesis are roughly organized according to the problem statement. It starts with the literature review of coaching, e-coaching and a suitable intervention technique for the e-coaching in chapter 2; then it moves to discussing multi-agent system and the realization of the e-coaching using the MAS paradigm in chapter 3. Next chapter 4 discusses what kind of conflicts can occur in MAS and how they are solved in relevant research studies. Chapters 5 and 6 integrate the two therapies in the system, and presents the conflicts resolution mechanism of the system respectively. Lastly chapter 7 concludes the thesis by discussing the overall success of the thesis and how this work can be extended in future.

Chapter 2

Coaching & E-coaching

In this chapter, we emphasize three things: i) the functionality of coaching, ii) cognitive behavioral therapy (CBT) as an effective intervention technique for mental and sleep disorders, and ii) how an e-coaching system can be modeled that can employ CBT as the intervention technique to coach the coachee on depression and/or insomnia. In this regard, in section 2.1, we illustrate coaching with all its criteria. Section 2.2 describes CBT. Section 2.3 discusses insomnia and CBT for insomnia (CBT-I) with all the techniques or exercises that it contributes to the treatment of insomnia. Similarly, section 2.4 discusses depression, CBT for depression (CBT-D) and the exercises that it offers to the treatment of depression. In section 2.5, we discuss the concept of e-coaching. And in section 2.6, we explain how e-coaching can be modeled in MHC. Finally, section 2.7 concludes the chapter.

2.1 Coaching

In general words, coaching can be defined as purposeful interactions between a coach and his coachees. A coach is someone who has expertise in a particular field, and he facilitates others to learn about that field and improve their performance in that area in a systematic way. For instance, Jeanne, a newly recruited real estate agent takes coaching from an expert real estate agent on sales negotiation skills. Or Roma, a yoga instructor coaches her clients on how to lead a healthy life through yoga.

Over the years, coaching has gained significant recognition in psychological research. As a result, now it is considered a newly emerging and applied sub-discipline of psychology [19]. In [20], it is defined as *the systematic application of behavioral science to the enhancement of life experience, work performance and well-being for individuals, groups, and organizations who do not have clinically significant mental health issues or abnormal levels of distress.*

From research studies, we can distinguish three types of coaching based on their purpose and scope: i) *executive coaching*, which is designed to help an executive coachee to improve her work efficiency in professional life [21, 22]; ii) *life coaching*, which is designed to enhance the life experience and goal attainment in the personal and/or professional life of a coachee [23]; iii) *personal coaching*, which is similar to life coaching, but it is based on empathy which helps a coachee in goal attainment, overcome obstacles and maintain motivation through self-understanding [24]. Among the three, life coaching has been seen a significant growth of interest among the psychologists in regard to practicing it professionally [25]. As a result, the Australian Psychological Society (APS) and the British Psychological Society (BPS) have acknowledged this interest by creating formal special interest groups in coaching psychology.

Life coaching can be done both in groups or individually. In any way, it should support an individual in three dimensions, i.e., i) goal-striving, ii) well-being, iii) hope. [23, 19].

- i **Goal Striving:** Life coaching especially emphasizes goal-striving and goal-setting. Since they are the foundation of self-regulation where an individual select his or her personal goals from the variety of life domains and work towards the goal-attainment [19]. It is found in research studies (i.e., [26, 27]) that the possession and progression towards life-goals increases the well-being of an individual. In addition, goals represent an individual's strivings to attain self-change, enhanced importance, and purpose in life [27].

- ii **Well being:** It refers to optimal mental functioning and experience [28] which involves the pursuit of happiness, pleasure, satisfaction of fulfillment, meaning and quality of life [29]. Through personalized feedback a coach can help a coachee to identify the actions or events of life that give her happiness, pleasure, satisfaction. For instance Jenny, a depressed person who isolates herself from others, finally gets to understand through coaching that she actually likes to be outside with others. Of course actions or events that affect the coachee negatively must be discouraged by the coach, for instance, Jenny enjoys smoking but it must be discouraged by the coach as it is not healthy for her.

- iii **Hope:** Hope is one of the important aspects that keeps an individual moving towards goals attainment. In order to achieve a goal, the coachee should think of the alternative pathways and she can only think of the alternative pathways and utilize them to reach the goal (agency), if she is hopeful that she will be able to reach the goal. According to [30], the integration of solution-focused, narrative and cognitive behavioral interventions are the best way to enhance one's hope.

In psychology, coaching is preferred over teaching or therapy perhaps because it involves a more collaborative approach where the coach can help the coachee through guidance and counseling; in addition the coachee's opinions are also respected [12]. In the coaching process, the coach should help the coachee to conceptualize clearer goals, think of numerous pathways to attainment, focusing the mental energy to maintain the goal pursuit, take obstacles as challenges to be overcome [31].

2.2 Cognitive Behavioral Therapy (CBT)

CBT is one of the most researched psychotherapy. Between 1986 to 1993, over 120 controlled clinical trials were included to the literature [32] and this proliferation still continues [33]. The literature of CBT includes over 325 outcome studies on different cognitive behavioral interventions. For its efficiency, and integrity it is being adapted for an increasingly wider range of disorders and problems [34, 35], i.e., depression, general anxiety disorder, social phobia, obsessive-compulsive disorder, posttraumatic stress disorder, schizophrenia, marital distress, bulimia nervosa, internalizing childhood disorders, sexual offending, chronic pain [36]. CBT holds the premise that how we think (cognition), how we feel (emotion), and how we act (behavior) are all interlinked. The core of this approach as pioneered by [37, 38] focuses on developing personal coping strategies by identifying and modifying unhelpful patterns in cognitions, emotions and behaviors.

There are two dimensions in the therapy: 1) *cognitive therapy*, which supports the coachee to identify and modify maladaptive cognitions; 2) *behavioral therapy*, which supports the coachee to identify and modify the problematic behaviors that can be self-destructive and unhealthy for her. The maladaptive cognitions include beliefs or schemas about the world, the self, and the future and they give rise to specific negative thoughts in particular situations, such negative thoughts are called *automatic thoughts*. For instance, Emma suffers from low-esteem. Her thoughts include "I am ugly," "I am stupid," and "no one likes to be with me"; related emotions are: shame, discomfort, grief, and depression; resultant behaviors include avoiding social interactions and isolating herself from others which make her disorder even worse. According to Beck's model, if Emma learns to identify and control her negative thoughts and replace them with positive thinking, she will be able to control her emotions, change her behaviors or actions and feel better. In contrast, faulty behaviors also affect one's cognitions and emotions, i.e., smoking or substance abuse are results of unhealthy and destructive behaviors which should be identified and modified for mental health.

Modern CBT by Donald Meichenbaum [39] acknowledges the quadratic reciprocity between the four domains of human experience: behaviors, thoughts, feelings, and resultant consequences. From a CBT viewpoint, the goal attainment becomes uncomplicated and accessible by understanding and structuring the relationships between the four domains in a way that they best support the process of goal attainment [23]. Hence, the therapy starts by providing psychoeducation to the coachee so that they can identify their agency in their emotional and behavioral experiences and develop skills to control how their thoughts, feelings, and behaviors interact and influence one another. Modern CBT refers to a class of interventions that combine a variety of cognitive, behavioral, and emotion-focused techniques e.g., [40]. The strategies emphasize cognitive factors, physiological, emotional, and behavioral components for the role that they play in the maintenance of the disorder.

CBT is evidence-based as it involves goal-directed, self-regulation process where a coachee sets a goal and plans actions required to achieve the goal, performs an action, evaluates his or her performances and gains insights from the consequences. In CBT, self-reflection is particularly important because it is assumed that it leads to insight and insight facilitates goal attainment and behavioral change [23]. CBT is considered a solution-focused approach because it is a humanistic approach which concentrates more on the strengths that the coachee brings to the therapy and emphasizes the importance of solution construction rather than problem analysis [23]. The overall goal of CBT includes symptom remission, improvement in functioning, and reduction of the disorder [41]. To accomplish the goal, the coachee has to become an active participant in a collaborative problem-solving process to examine and challenge the validity of maladaptive cognitions and to change maladaptive behavioral patterns.

In CBT, the role of the coach is important because he can play a role to nurture and maintain a sense of hope in the client's mind [19]. In [39], CBT is viewed as an integrative, biopsychological intervention technique which entices attention to many previously unacknowledged aspects like the coachee's social and personal lives' context, past episodes, their strengths, and resources. Hence, the coach needs to create and maintain a genuine, empathic, nonjudgmental and supportive therapeutic relationship with the coachee. The relationship between the coach and the coachee support the coachee's primary therapeutic goals, i.e., developing coping skills, taking credit for changes and relapse [39].

2.3 Insomnia & CBT for Insomnia (CBT-I)

Insomnia is a sleep disorder, also known as sleeplessness where people face trouble sleeping which include initiating and/or maintaining sleep. It is a very prevalent complaint which can exist as a primary complaint or in association with another physical or mental health complaint [42]. Prevalence estimates show that about one-third of the adult population reports insomnia symptoms and 9-12% also suffer from additional daytime consequences [43]. It causes rigorous individual and societal consequences, for instance, concentration problems, increased risks of accidents, depression, reduced productivity, increased absenteeism [42]. Persistent insomnia which lasts for more than a month or sometimes a couple of years is associated with increased risk of depression and chronic hypnotics [44, 45, 46, 47].

The diagnosis of insomnia includes the subjective complaint of difficulties falling or staying asleep. The indicators to quantify the severity and clinical significance of insomnia include time to fall asleep, duration of awakenings, total sleep time, frequency and duration of sleep difficulties.

Research studies show that CBT-I generates sustainable positive changes in the condition of insomnia [48]. CBT-I offers a 6 to 10 weeks coaching session along with a variety of exercise types that are different in aim and properties, i.e., sleep restriction, stimulus control, relaxation, cognitive therapy, and sleep hygiene. The coach may advice the coachee to practice some of the exercises on a regular basis to get a good night sleep and prevent insomnia, for instance, sleep hygiene, relaxation before sleep.

- **Stimulus Control:** Stimulus control exercise is one of the most effective techniques for insomnia. According to the technique, the insomniac coachee should only use the bed when she is sleepy, and the bedroom only for sleep and sex. After going to bed, if she does not fall asleep within 15-20 minutes, she should go to another room and repeat the process. The coachee is advised to maintain a sleep diary to register everyday wake up time and quality of the sleep. She should also avoid day time napping. [49, 50].

- **Sleep Restriction therapy:** This exercise limits the hours spent in bed each night to the actual amount of sleep time. If the insomniac coachee complains that she sleeps an average of 6 hours per night out of 8 hours spent in bed, then the initial sleep window (from lights out to arising time) would be restricted to 6 hours. This would initially create sleep loss and make the coachee even more tired. But it would make the coachee fall asleep faster and wake up fewer times in the night. As her sleep improves, the restriction will be reduced. When the sleep window is reduced to the number of hours she needs without diminishing the sleep quality, the goal of this exercise will be achieved [51].

- **Relaxation:** It teaches the coachee to relax both her mind and body. It also helps her to reduce any anxiety or tension that causes her stay awake at night. The goal of this exercise is to train the coachee on how to control muscle relaxation, breathing, mental focusing.

- **Cognitive Restructuring:** Cognitive therapy involves challenging and changing the misconceptions and beliefs about insomnia and its perceived daytime consequences.

- **Sleep Hygiene:** This exercise helps the coachee to correct the things or habits that disturb her sleep, i.e., caffeine, long napping in the afternoon. It also guides the coachee to develop a pattern of healthy sleep by changing sleep position, rearranging the bedroom, setting a comfortable temperature, soothing light, or music.

2.4 Depression & CBT for Depression (CBT-D)

Major depressive disorder (MDD) or depression refers to a mental state of low mood which significantly affects an individual's behavior, thoughts, feelings, emotions, and health. Depression is one of the primary reasons of suicide and nearly 800,000 people die to suicide every year [52]. In the United States, depression is the key cause of affliction for ages between 15 and 44 [53]. Moreover, it is projected as the leading cause of disability and the second leading contributor to the global burden of disease by 2020 [54]. Diagnostic and Statistical Manual of Mental Disorders, 4th Edition (DSM-IV) specifies 9 diagnostic criteria for depression [55] which include: loss of interest or pleasure in doing things, feeling depressed, insomnia or hypersomnia (sleeping too much), tiredness, a significant change in appetite, feelings of guilt, lack of concentration, psychomotor retardation (slowing down in thinking or physical movements), and having suicidal thoughts.

Research studies show that with adequate psychopharmacological interventions, cognitive behavioral therapy for depression or CBT-D can be found to be equivalent in efficacy to the antidepressant medications (ADM) in the treatment of depression [56, 57, 58, 59]. In fact, the effect of the ADM does not persist after the completion of the treatment, but the effect of CBT-D persists even after the completion of the treatment [60, 61, 62, 63]. The one-year relapse-rate of CBT-D is approximately 30% whereas the one-year relapse-rate of ADM is approximately 60% [64]. CBT-D offers a variety of cognitive and behavioral exercises, i.e., activity scheduling, cognitive rehearsal, self-reliance, role playing, role reversal, distraction techniques, briefing, thoughts recording, investigation, reattribution, and alternative solutions. The exercises are planned and scheduled based on the coachee's current mental health condition. Furthermore, once the coachee understands the reasons behind her depression, and can handle her automatic thoughts and attitudes on her own she may not need to continue the exercises. However, she should have the complete knowledge about the exercises so that in future she can practice them for relapse prevention.

- **Activity Scheduling**: The scheduling of activities is frequently used in the early stages of treatment to counteract the loss of motivation, hopelessness, and excessive rumination. The depressed coachee maintains an hourly-record of the activities in which she has engaged recently and rate each completed activity on fulfillment and pleasure. The purpose of this exercise is getting into the situations where the ratings would contradict her beliefs that she cannot accomplish or enjoy anything anymore. These contradictions will eventually make the coachee realize her automatic thoughts that are triggered in her mind regardless of the situations. If it gets difficult for the coachee to accomplish an activity, the coach subdivides it into segments, ranging from the simplest to the most difficult and complex aspects of the activity. It is called "graded task" approach, and it enables the coachee to undertake tasks that were initially impossible, but at the end provides proof of success.

- **Cognitive Rehearsal**: Cognitive rehearsal involves asking the coachee to imagine each step involved in the accomplishment of a particular activity. This technique can be especially helpful with those who have difficulty in carrying out an activity that requires successive steps for its completion. The imagery evoked by the cognitive rehearsal technique helps both the coach and the coachee. The coachee learns how to focus, and the coach identifies obstacles that make an activity difficult for the coachee.

- **Self Reliance**: Some depressed coachees rely on others to take care of most of their daily needs. Through self-reliance training, the coach explains the importance of trivial daily works such as showering, making beds, cleaning the house, cooking meals, shopping, and asks the coachee to follow a daily routine to complete all these tasks. The coach may also help the coachee to complete the activities by notifying her about them.

- **Role Playing**: This technique is used to bring out automatic thoughts through the enactment of particular interpersonal situations, such as an encounter with a supervisor at work. It can be used to guide the coachee on how to exercise and organize new cognitive responses and behaviors in problematic social encounters.

- **Role Reversal**: Role reversal can be very effective in helping the coachee to realize how others might view her behavior. Through role reversal, the coachee begins to view herself less harshly as "self-sympathy" responses are evoked.

- **Distraction Techniques**: The coach can introduce various distraction techniques to assist the coachee to reduce the intensity of painful effects. The coachee learns how to divert negative thinking through physical activity, social contact, work, play and visual imagery. Practicing diversion techniques also helps the coachee to gain further control over emotional reactivity.

- **Briefing**: When the coachee is able to identify external events and situations that evoke particular emotional responses, the coach can use imagery technique in weekly briefings by asking the coach to relax, close her eyes and picture a distressing situation which she encountered or often encounters in detail. The coach describes what has happened in the past as she relives the event. If the distressing event is an interpersonal one, the coach plays the role of the other person in the encounter, while the coachee plays herself. The automatic thoughts can usually be evoked when the coachee becomes sufficiently engaged in the role-play. In attempting to elicit automatic thoughts, the therapist is careful to notice and point to any mood changes that occur during the session.

- **Thoughts Recording**: Once the coachee becomes familiar with the techniques for identifying automatic thoughts, she is instructed to keep a daily-record of dysfunctional thoughts, in which she records the emotions and automatic thoughts that occur in upsetting situations. The coachee is motivated to develop rational responses to her dysfunctional automatic thoughts listed in the daily record.

- **Investigation**: When the coach and the coachee have managed to isolate a key automatic thought, they approach the thought as a testable hypothesis. Through the procedures of gathering data, evaluating evidence, and drawing conclusions, the coachee learns that her view of reality is different from what it actually is. In testing automatic thoughts, it is sometimes necessary to refine the coachee's use of a word, i.e., instead of using "bad," "stupid" or "selfish," the coachee has to be specific and definite about her expressions. For instance, if a depressed student says he failed in math, it may sound too general to the listener. But if the student says he was unable to achieve C grade after spending so much time as an average student, it becomes more clear and understandable for the therapist to examine past evidence and test the validity of the hypothesis. This process can help the coachee to see the over-inclusiveness of her negative self-assessment and the idiosyncratic nature of many automatic thoughts.

- **Reattribution**: Another useful technique is reattribution which trains to reject incorrect and self-blaming thoughts. It can be used when the coachee unrealistically associates adverse occurrences to a personal deficiency, such as lack of ability or effort. The coach and the coachee review the relevant events and apply logic to the available information to make a more realistic assignment of responsibility. The aim of reattribution is not to exempt the coachee from all responsibility but to examine many factors that contribute to adverse events. Through this process, the coachee gains objectivity, relieve themselves from the burden of self-blame, and searches for ways to solve problems or prevent their recurrence. The coach also uses reattribution technique to show the coachee that some of her thinking or behavioral problems can be symptoms of depression (e.g., loss of concentration) but no signs of physical decay.

- **Alternative Solutions**: When the coachee becomes capable of identifying and modifying automatic thoughts; and understand natural life problems, the coach trains her to look for more suitable solutions by generating several alternative solutions of a problem. Because a depressed person's reasoning often gets limited, an effort to reconceptualize a problem can result in detecting a viable solution that the coachee previously might have rejected.

2.5 E-coaching

Delivering coaching via traditional means, i.e, face to face interactions with a coach, is expensive and difficult to scale up [65]. Also, with technological evolution when people prefer to receive services at home, a person physically going to a clinic seems quite a bit of bother. Hence, the concept of an e-coaching system has become sensational as it changed the whole idea of having face to face coaching. E-coaching is defined as any form of coaching that takes place using electronic media with or without the input of a real coach [66]. In e-coaching, a human coach interacts with his or her clients in two ways: indirect and direct. In the indirect communication, the coach records all types of coaching materials and pieces of advice in a video, audio or text formats and the coach has to go through the records to find solutions of her problems; this approach is called *self-guided coaching*. On the other hand, in the direct communication, the coach directly communicates with the coachee through the electronic media, i.e., Internet, text message, or call; this approach is called *guided coaching*.

With advancements in information technology (IT) and Artificial Intelligence (AI), attempts are being made to replace the human coach with an automated system and make it a fully-automated e-coaching system. Research studies show that a non-human, computer embodied agent or e-coach can provide a successful coaching by making a working relationship of trust and adherence with its human clients [67]. In this attempt, several health coaching dialogue systems have been developed which utilize persuasive technology and behavior medicine to provide treatments ranging from depression to insomnia [68, 69, 70, 71].

A fully automated e-coaching system offers many advantages over a human-operated e-coaching system, like the constant connection between the coach and his coachees which supports momentary feedback, consultation on the current context and problem sharing at any time. Also, the infrastructure of such a system enables integration of objective data into the treatment after obtaining them from the non-obtrusive sensory measurement, i.e., pedometers for obesity, wearable [72] for mood disorders. Information exchange between user groups, i.e., peers, human therapists, and medical institutions also enhance the performance and treatment policy of the system. One of the significant features of a fully automated e-coaching system is that it can also be a part of a stepped health-care structure which means if the health of a client declines or other serious problems arises in the treatment procedure, human coaches can take over the system.

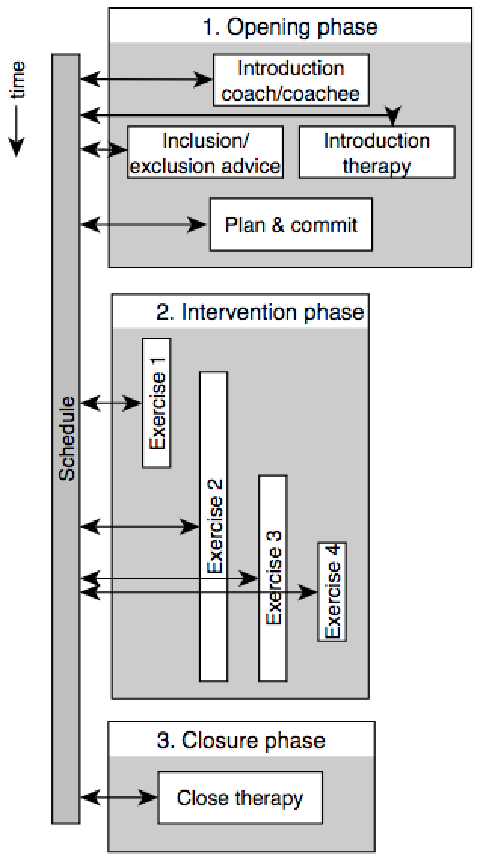

Figure 2.1: A general model to show the coaching process of MHC. Source: [9].

2.6 Modeling E-coaching

We assume that the coaching on depression and/or insomnia may take approximately 6 to 10 weeks (the coaching of insomnia takes 6 weeks. However how long the depression coaching may take is undetermined). The coaching includes a number of assignments and consultation or briefings based on exercises types. The coaching process consists of three phases: 1) *opening phase*, 2) *intervention phase*, and 3) *closure phase* (see Figure 2.1). The goals of the opening phase are improving transparency through the process of alignment and establishing commitments to the coaching process in the form of agreed contracts or agreements. Alignment may indicate sharing information at three different levels: *therapy level*, *communication level*, and *ethical level*. In the therapy level, the system conveys therapy related information and specifications of the requirements to the coachee, i.e., how much time and effort must be invested, and what are the goals of the coaching process. The communication level alignment may refer to providing the coachee a personalized interface (i.e., a female coachee may prefer pink color in the interface, etc.). Privacy assurance, building trust with each other and risk management are parts of the ethical level alignment which can be achieved by signing contracts. In Figure 2.1, we can see that to build alignment and commitment to the coaching program, 4 types of activities are scheduled in the opening phase: 1) *introduction of the coach and the coachee*; 2) *introduction of the therapy*; 3) *inclusion and exclusion advice*, and 4) *plan and commit.* At the beginning of the opening phase, the coach and the coachee introduce themselves to one another; the coach also introduces and explains the coaching program to the coachee. The coachee explains her problems (disorders), mentions the symptoms she is suffering from and what she expects from the coaching. After hearing the coachee, the coach decides whether the coaching program is suitable for her disorders. If the coach thinks the coachee's disorders are too severe to be treated through the e-coaching, he will suggest her not to join the coaching (exclusion advice); Otherwise he will include her in the system as the current coachee. After including the coachee, the coach and the coachee sign agreements or contracts on following the constraints of the system.

The intervention phase consists of cognitive and behavioral exercises which are planned to support the coachee restoring her mental health and help her to achieve the goals. The starting and (estimated) ending times of the planned exercises are entered into *Schedule* (a shared tool between the coaching participants which is used to schedule the exercises). Planning of exercises may take place during the briefing sessions where the coach and the coachee discuss the ongoing coaching process, the coachee's progress or decline and further exercises. Exercises for the first week or first few weeks can also be planned during the opening phase. In some cases, an exercise may have prerequisites of finishing some other exercises. In that case, that exercise will be scheduled when its prerequisites exercises are finished, for instance, the CBT-D exercise of investigation of automatic thoughts may have the prerequisite of performing throughts recording exercise. So until the coachee performs thoughts recording exercise, investigation exercise will not take place. The coach communicates with the coachee before the starting of a new exercise so that they can introduce and explain the exercise with all its constraints.

Each exercise consists of four steps: A) *introduction*, B) *plan and commit*, C) *task execution* and D) *evaluation* (see Figure 2.2). In step A, the exercise is introduced to the coachee; in step B, the coach explains the constraints that the coachee has to follow during the coaching and expectations that they have from the coachee in that exercise. The coachee can negotiate about the specifics of an exercise (i.e., starting or ending time, etc.). When the coach and the coachee agree, they sign a contract as a proof of their commitments to the exercise. In step C, the coachee performs the exercise and in step D, the coach and coachee participate in a briefing session where they evaluate the coachee's performance based on her adherence and performance data. After evaluation, either the exercise ends or continues (redo) along with discussing plans for the next cycle of exercises. When all the exercises are performed, the closure phase starts to indi-

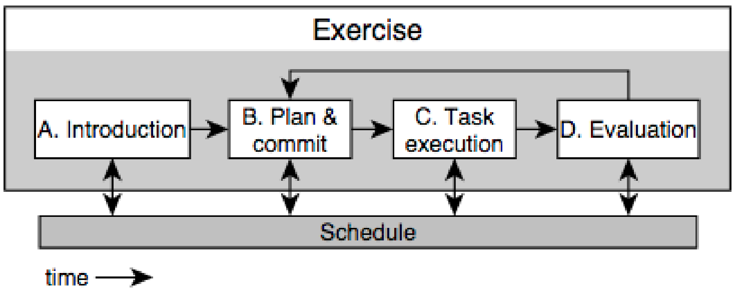

Figure 2.2: Interaction model of an exercise [9].

cate the ending of the coaching process. It can start even before completing all exercises if the coachee wants to withdraw from the coaching or if the coach advises the coachee to withdraw from the coaching based on the coachee's health. In the closure phase, the coach and the coachee participate in evaluating the coachee's progress, and plan relapse prevention exercises which the coachee can perform on her own after the coaching. The coaches may also provide valuable information and supportive strategies on how to live a healthy life mentally.

2.7 Conclusion

In this chapter, we have discussed the basic concepts that are required to model the e-coaching system, i.e., coaching, cognitive behavioral therapy as a coaching intervention technique, and e-coaching. Coaching can be defined as purposeful interactions between a coach and his coachees. A coach is someone who has expertise in a particular field, and he facilitates others to learn about that field and improve their performance in that area in a systematic way. Over the years, coaching has gained significant recognition in psychological research. As a result, now it is considered a newly emerging and applied sub-discipline of psychology [19]. CBT is a very popular intervention technique for mental and behavioral disorders like depression, general anxiety disorder, social phobia, obsessive-compulsive disorder, posttraumatic stress disorder, schizophrenia, marital distress, bulimia nervosa, internalizing childhood disorders, sexual offending, chronic pain [36]. It holds the premise that how we think (cognition), how we feel (emotion), and how we act (behavior) are all interlinked. There are two dimensions in the therapy: 1) *cognitive therapy*, which supports the coachee to identify and modify maladaptive cognitions; 2) *behavioral therapy*, which supports the coachee to identify and modify the problematic behaviors that can be self-destructive and unhealthy for her. Insomnia is a very prevalent sleep complaint which can exist as a primary complaint or in association with another physical or mental health complaint [42]. Whereas, major depressive disorder (MDD) or depression refers to a mental state of low mood which is considered one of the primary reasons of suicide. E-coaching is defined as any form of coaching that takes place using electronic media with or without the input of a real coach [66]. With advancements in information technology (IT) and Artificial Intelligence (AI), attempts are being made to replace the human coach with an automated system and make it a fully-automated e-coaching system. A fully automated e-coaching system offers many advantages like the constant connection between the coach and his coachees which supports momentary feedback, consultation on the current context and problem sharing at any time. Based on these concepts, we have described how e-coaching can be modeled in MHC. The coaching process of MHC consists of three phases: 1) opening phase, 2) intervention phase, and 3) closure phase. The goals of the opening phase are improving transparency through the process of alignment and establishing commitments to the coaching process in the form of agreed contracts or agreements. The intervention phase consists of cognitive and behavioral exercises which are planned to support the coachee restoring her mental health and help her to achieve the goals. In the closure phase, the coach and the coachee participate in evaluating the coacheeâĂŹs progress, and plan relapse prevention exercises which the coachee can perform on her own after the coaching.

Chapter 3

Multi-Agent System for E-coaching

In this chapter, we give a literature review of an agent and multi-agent systems. Furthermore, we show how we can model an e-coaching system using the multi-agent system paradigm. In this process, section 3.1 defines what an agent is, and section 3.2 explains how a multi-agent system is composed of multiple agents and why it is considered an attractive paradigm for application developments. Section 3.3 discusses the essential properties of a multi-agent system, i.e., communication and cooperation between agents. In section 3.4, we model the basic structure of a coach. Section 3.5 explains how we can model an e-coaching system using MAS. Finally, section 3.6 concludes the chapter.

3.1 Agent

In Computer Science (CS) and Artificial Intelligence (AI), the term agent has been used in various ways, i.e., the software agent, intelligent agent, autonomous agent. Simply, an agent can be viewed as an "Electronic Assistant" which assists its user by acting on behalf of him/her. So far, there is no unanimously accepted answer to the question of what an agent is. However, the most popular definition is given by Wooldridge and Jennings in [73]; according to them, an agent can be defined as a hardware or software-based computer system which holds the following properties.

- **Autonomy**: An agent is autonomous if it can operate without the direct interventions of humans or other components in the system. It is the most fundamental property of an agent which gives it a degree of independence to react to its dynamic environment in the absence of humans and others.
- **Social ability**: An agent can interact with other agents and humans via some communication mechanism. This property becomes essential when

multiple agents collaborate to work on something.

- **Reactivity**: Besides having other agents as colleagues within the system, an agent also has an external entity, its environment in which it is situated. An agent should have the ability to comprehend its environment and react to it in a timely fashion.
- **Proactivity**: It refers to the idea that an agent should not only be able to react to its environment, but it should also be able to exhibit goal-directed behaviors by taking initiatives.

Getting inspiration from the intentional stance of [74], agents are often characterized in terms of mental notions or properties like *knowledge*, *belief*, *desire*, *obligation*, and *intention* and such agents are termed as cognitive agents. It is argued by Dennett that such low-level descriptions may not be suitable to explain and predict a complex system's behavior. However, such mental properties provide a useful and valid abstraction of the system, perhaps a multi-agent system and how it works.

The most popular architecture for cognitive agents is the BDI-architecture [75], which characterizes an agent in terms of *beliefs*, *desires*, and *intentions*. Beliefs express the informational states of the agent about the world including itself. Desires express the motivational states of the agent which includes objectives or situations that the agent wants to achieve. The difference between a goal and a desire is that a goal is a desire that has been adopted for active chase by the agent. Intentions express the deliberative state of the agent, what the agent has chosen to do. An intention is those desires to which the agent has some kind of commitments.

3.2 Multi-Agent System

A multi-agent system (MAS) can be defined as a group of entities who interact with each other to achieve individual or collective goal [76, 6]. Nowadays, the MAS paradigm is being used for a wide range of entertainment and business applications such as games, movies, geographical information system (GIS), transportation, and logistics. The use of MAS is also increasing in healthcare domain for different purposes, for instance, patient appointment scheduling, i.e., [77, 78], patients' information management, i.e., [79, 80], diagnosis and decision support systems, i.e., [81, 82, 83], and organ and tissue transplant management, i.e., [84, 85, 86]. Besides, recently MASs are also being used to provide personalized healthcare to different groups of patients for different purposes such as senior citizen care [87, 88, 89], medical training and education [90], insomnia treatment [91]. Initially, it was assumed that MAS is so popular because of its computational power and advantages in development and maintenance. However, the truth is that the new additions in the MAS paradigm that is open multi-agent system (open MAS) enables us to realize a software system in a way that is unprecedented in the history of software engineering [92]. Below, we discuss the advantages of the MAS paradigm.

- **Distributed problem-solving**: MASs utilize distributed problem solving, which has many advantages such as fast parallel computing, flexibility through partitioned expertise, narrow-bandwidth high level communication, and increased fault tolerance [76].
- **Modularization**: The paradigm modularizes a system into multiple independent, autonomous entities or agents. The modularization into autonomous agents greatly expedites the development and maintenance of a software system. The significance of this benefit is realized when a team of developers is involved in the realization of a MAS. The developers can individually focus on one or more agents without worrying how their agents will fit into the overall system which reduces the development time as the team does not need to spend time to decide how to tune their designs [92]. In the maintenance phase, the MAS paradigm is especially helpful because it enables the developers to adjust one component or agent of the system while not altering rest of the system. Moreover, the system can be updated easily by including new agents or altering only the agent(s) which needs to be updated.
- **Open MAS**: An open MAS provides a receptive setting for the agents where they can enter or leave the system dynamically. Which agents or how many agents will constitute a system need not to be known in advance. Every developer can contribute to the system by adding his or her agent(s) in the system. Examples of open multi-agent systems are electronic systems, i.e., electronic auction houses [93].

3.3 Properties of a MAS

Generally, MASs are employed to accomplish the problems that are unlikely or difficult for a single agent or a monolithic system to accomplish, i.e., autonomous driving. The process of multiple agents working on smaller parts and thus accomplishing the bigger problem is only possible if they are able to interact with each other. For instance, in a healthcare application there can be several agents with distinct responsibilities who can fulfill their responsibilities independently (i.e., organizing and managing medical information, monitoring the current condition of the patient, giving reminders or notifications of his medications). However, to achieve something which is not possible for a single agent to achieve alone (i.e., comparing the effects of current medications to the impact of previous medicines), they must cooperate with each other and work as a team by coordinating actions and share information and knowledge through communication.

3.3.1 Agent Communication

Communication is not a new concept in CS or IT. According to Shannon's theory of communication [94], a computer (connected to the printer) sending a print instruction to the printer, objects interacting each other through method calls in object-oriented programming (OOP), users interacting with the application through graphical user interfaces (GUI) are also considered as communication. However, in all of these instances, the receivers have no authority to make the decision of what to respond to the sender, i.e., a GUI button does not have any authority to make the choice of returning "OK" or "Cancel" as the developers have already determined its response. Such communications are termed as *deterministic communication* in [92]. But, in AI, autonomy is an essential property of an intelligent agent and it is expected in communication also. This suggests that the receiving agent should make the decision on how or whether to react to a communicative activity.

The agent communication has garnered much attention in research over the last decade, and human communication has inspired much of its research. In a MAS, the agents can automatically form a cooperating unity if there is an interaction mechanism known as agent communication language (ACL) [92]. An ACL supports interoperability in a system if all the agents of the system ascribe the same meaning to the language, in other words, if they are semantically interoperable. Semantically interoperable agents can understand each other's message but are free to react to them as they please. An ACL establishes a *non-deterministic communication* in a MAS since it does not determine the behaviors of the communicating agents [92]. Popular agent communication languages include *KQML* [95], *FIPA-ACL*[96], and *KQML Lite* [97].

3.3.2 Cooperation

Cooperation is a key MAS concept [13, 14, 15, 16] which is implemented for four major goals [17] : i) improving the rate of task completion through parallelism; ii) increase the number of parallel tasks by sharing information and resources; iii) increase the chances for tasks completion by reusing the plan or outcome of previously done similar tasks or realizing the task from different modes; iv) eliminate negative interactions between tasks. Nevertheless, cooperation is still considered a highly complex and inconsistent procedure [98] which draws many other serious problems like conflicts or disagreements between agents. For instance, two or more agents may have incompatible goals that cannot be achieved collectively; in such a situation, the agents may get involved in a conflict regarding whose goal should be chosen to be accomplished (more details in chapter 4).

In MASs, cooperation can range from total cooperation to total antagonism [99]. In a total cooperative system, agents may change their goals to accommodate the requirements of other agents. In contrast, in a total antagonistic system, agents compete and may create hindrances for others in the way of their goal achievements. Hence, a MAS can be characterized by the type of cooperation it implements. In either type of systems, conflict is a serious problem which needs to be resolved according to the system architecture to get the expected outcome of cooperation. We will learn more about conflicts in the next chapter. However, in the next section, we describe how we can model the core entity of the MHC that is a coach that will be able to provide e-coaching instead of a human.

3.4 Modeling a coach

We model a coach as an intelligent autonomous agent who holds some of the essential properties of a human coach, i.e., the coach should have expertise and knowledge in a particular domain; he should be able to comprehend the coachee within particular situations and circumstances; he should be able to offer an effective coaching program to the coachee based on her current condition; and he should be able to motivate and encourage the coachee in her process of behavior change.

We initiate to model the coach using the BDI architecture [75] where beliefs of the coach are built on his domain knowledge; the information about the current coachee, and the ongoing coaching process. We coin *CoModel* as the knowledge-base/repository of the coach that contains these three types of beliefs or information in three different repositories:

- **Background**: It contains a lot of "timeless" information about the general coaching process such as disorders (symptoms), techniques, exercises etc. In other words, *Background* contains the background knowledge of the coach.
- **CoProcess**: It contains all officially recorded, time-indexed information about the current coaching process, e.g., information about the activities that are done, information about the plans for the future, records from the intakes, diagnosis, and the coach's views on the overall state of the coachee in reflection to the therapy.
- **CoActual**: It reflects information regarding the current coachee. The information in this repository is seen and maintained by the coachee. For instance, name, birth date, gender, current situation that she is in.

The coach has a stable *desire* or *goal* that is treating the coachee (in this context, we assume desire and goal can be considered interchangeable). In order to fulfill the *desire*, the coach models a set of *intentions* in the form of a set of intended actions or plans. The coach plans his intended actions locally on a tool named *local_schedule*. This tool comprises of the the coach's primarily intended therapeutic techniques or exercises, activities, and events along with their starting and ending times. Actions of the *local_schedule* are not the final outline of the coaching program because if the coach resides in a cooperating environment with other coaches, then he may need to share his local plans from the *local_schedule* with them. Also, in any case the coach cannot finalize a plan until he shares it with the coachee, and the coachee agrees with the coach's plan. It gives the coachee autonomy to bargain about the times or specifics of any entries of the *local_schedule*.

The coach shares his local plans with other coaches of the system and the coachee via a tool named *Schedule* where all the therapeutic exercises with all their specifications are entered as the final outline of the coaching program. If there are multiple coaches in the e-coaching system, everyone has their own local plans for the coaching program, then they communicate their plans by sharing *local_schedule* and finalize the final plan via negotiation and modification on the *Schedule*. An activity scheduling mechanism is followed to plan exercises for the *Schedule*. We will learn more about it in chapter 6.

The *Schedule* tool is attached to *Contracts*. A *Contract* is a list of *Constraints* which can be rules, conditions or commitments that the coachee and coach have agreed to follow during the coaching process by signing it. In other words, *Constraints* are the norms of the system that make sure that the coaching process is acceptable and conforms to the proper standards. When exercises are entered on the *Schedule* to share with the coachee that time the relevant Contracts are also integrated on the *Schedule*.

There are two types of *Constraints*: behavioral constraints, and informational constraints. The behavioral constraints impose restrictions on the coachee's habits or life routine which may hinder the schedule of the coaching process (e.g., dinner should proceed by at least 3 hours before going to bed). Various *Contracts* or agreements that the coachee or coach enter into can be seen as behavioral constraints. The informational constraints regulate the quality of the communicative actions. They ensure that the coachee is informed when a certain condition arises, or they may prevent the coach from repeating itself, i.e., providing feedback on an exercise. A *constraint based approach* is followed to make sure that both the coaching bodies are acting according to the *Constraints*.

If there is a violation of the *Constraints* which is referred to conflict, the coach repairs it by generating dialogues or *dialogue_actions*. There is a process called *Negotiator* which selects interaction recipes (*Interaction_recipe*) from *Background* to generate *dialogue_actions* in regard to resolve the constraint violation.

There are a number of *dialogue_actions* between the coachee and the coach that update the *CoProcess* of the coach and change the state of the coachee. The coach has an *Inventory* of templates for all the possible *dialogue_actions* that can happen on behalf of the coachee regarding his particular domain and a dia-

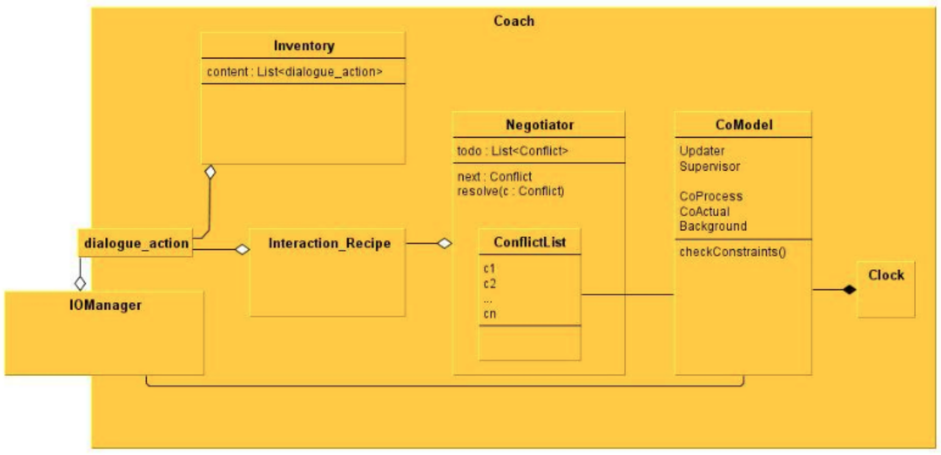

Figure 3.1: Internal architecture of a coach. Source: [10]

logue generator process to generate *dialogue_actions* according to the situations. The *dialogue_actions* are an essential part of the coaching process and any of the coaching party can initiate conversation through them. The main uses of *dialogue_actions* include:

- They are the basic communicative actions and the building blocks of more elaborate *Interaction_Recipes*,i.e., introduction, explanation, consultation;
- They are also used in the planning and scheduling of exercises, and making changes in the plan or schedule; the coachee can also have *dialogue_actions* with the coaches during performing exercises as well if she wants to;
- The e-coach may also use *dialogue_actions* to influence other aspects of the coachee, like, for instance, trust, motivation, uncertainty, or stress;
- *dialogue_actions* may change the status of scheduled events or exercises if they contain the information that an exercise has already been done, or canceled, or led to certain performance scores;
- *dialogue_actions* may contain information about progress or impediment/delay of the underlying coaching process;
- They may trigger a reaction from the system in order to, for instance, comment on this progress, or to try to overcome impediments.

Besides the three repositories, *CoModel* also has two active processes which keep the *CoModel* updated, and help the coach to comprehend the dynamic environment.

- *Updater*: It updates *CoProcess* based on the information coming from *CoActual* and *dialogue_actions*, e.g., the times change of a scheduled exercise.

- *Supervisor*: This process looks for violations of the *Constraints* by comparing what the coaching bodies are doing and what they are supposed to do according to the Contracts.

The internal architecture of the coach is given in the Figure 3.1. The figure shows that there are *CoModel* which contains knowledge bases for the coachee and is connected to the system clock to keep the model up to date. Tools like *local_schedule*, and *Contract*s, *Constraints* are part of the CoModel. Inside the CoModel, there is also a method called *checkConstraints*() which keeps a check on the coachee to see if she is following the *Constraint*s that she agreed to follow. If any conflict is detected, it sends it to the *Negotiator* process. The Negotiator generates suitable dialogue_actions using the *Inventory* and *Interaction_Recipe* to resolve the *Conflict*. And it communicates the dialogues to the coachee via the *IOManager*.

3.5 Multi-Agent System for E-coaching

On the top level, there are three components: a coachee, an e-coaching system, and an environment. Inside the e-coaching system, there are four agents[1]: i) DepressionCoach, an expert in depression domain; ii) InsomniaCoach, an expert in insomnia domain; iii) IOManager, an input-output manager who manages the interactions between the coaches and the coachee; and iv) Scheduler who integrates the local plans of the coaches and schedule them on the Schedule after executing the conflict detection and resolution processes on them (we will know more about conflicts and conflict detection and resolution in the next chapter, and the tasks of the Scheduler agent are described in chapter 6). The symptoms that the coachee is suffering from decide between the two coaches who will coach her. For instance,

I If the coachee has only depression symptoms (i.e., low mood), DepressionCoach will conduct the coaching, and the InsomniaCoach will be inactive (see Figure 3.2).

II If the coachee has only insomnia symptoms (i.e., sleep disturbances), InsomniaCoach will conduct the coaching, and DepressionCoach will be inactive (see Figure 3.3).

[1]We use "agent" term to refer to all the agents of the system, and "coach" to refer to the coaches, i.e., DepressionCoach and InsomniaCoach. And by the term "e-coach" or "e-coaching system" we refer to MHC

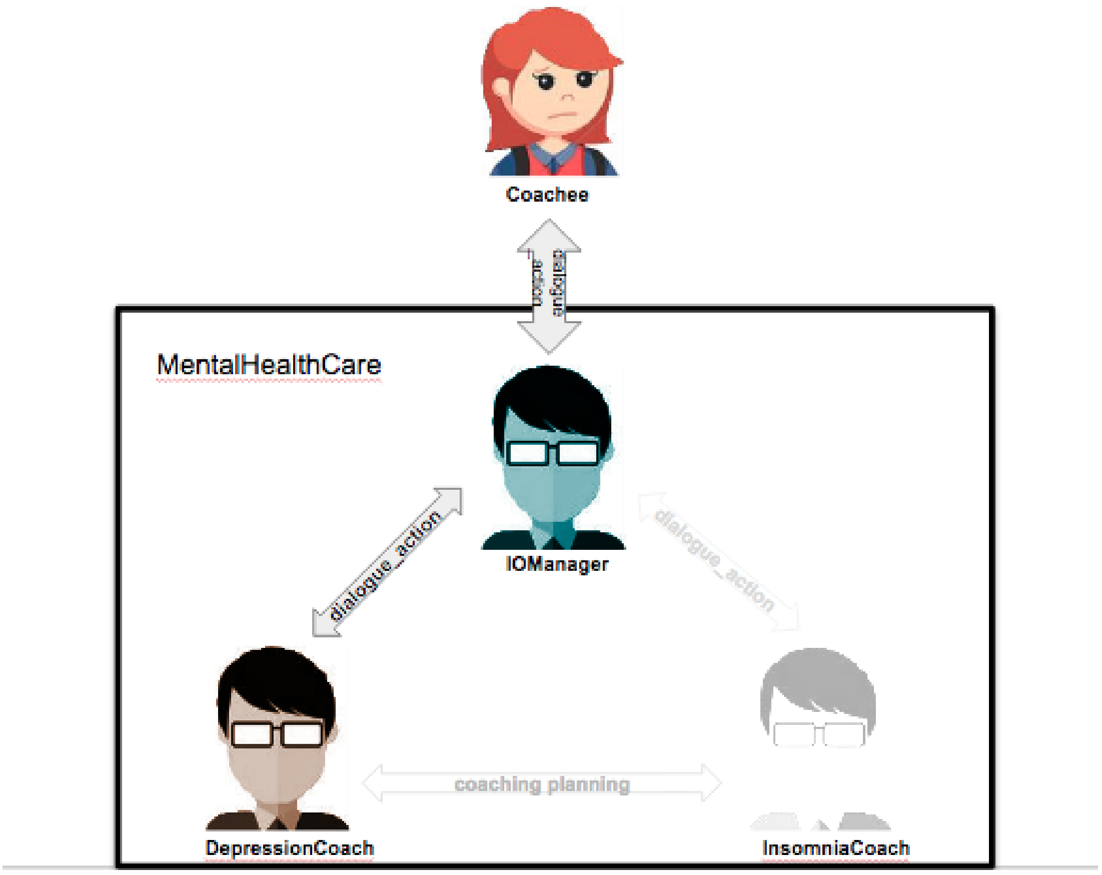

Figure 3.2: Interaction model of the depression coaching

III And if the coachee has symptoms of both depression and insomnia, the coaches will work cooperatively to offer the combined coaching on depression and insomnia (see Figure 3.4).

In order to facilitate the coachee with the combined coaching, the system must ensure two things: communication, and coordination between the agents so that they can cooperate with each other. For communication, inside the system, there are interaction channels between the agents through which they are connected and communicate with each other. The coaches (DepressionCoach and InsomniaCoach) may also share interaction channels with the outside sensor devices (if there is any) through which they monitor the coachee and her surrounding, i.e., wearables or smart watch to know about the coachee's pulse. All information passing through various channels inform the connected coach(es) about the coachee and her state.

We can identify two types of interactions that occur between the system and the coachee: *indirect interaction*, that occurs via the channels with the sensor devices to receive information about the coachee from her environment, and ii)

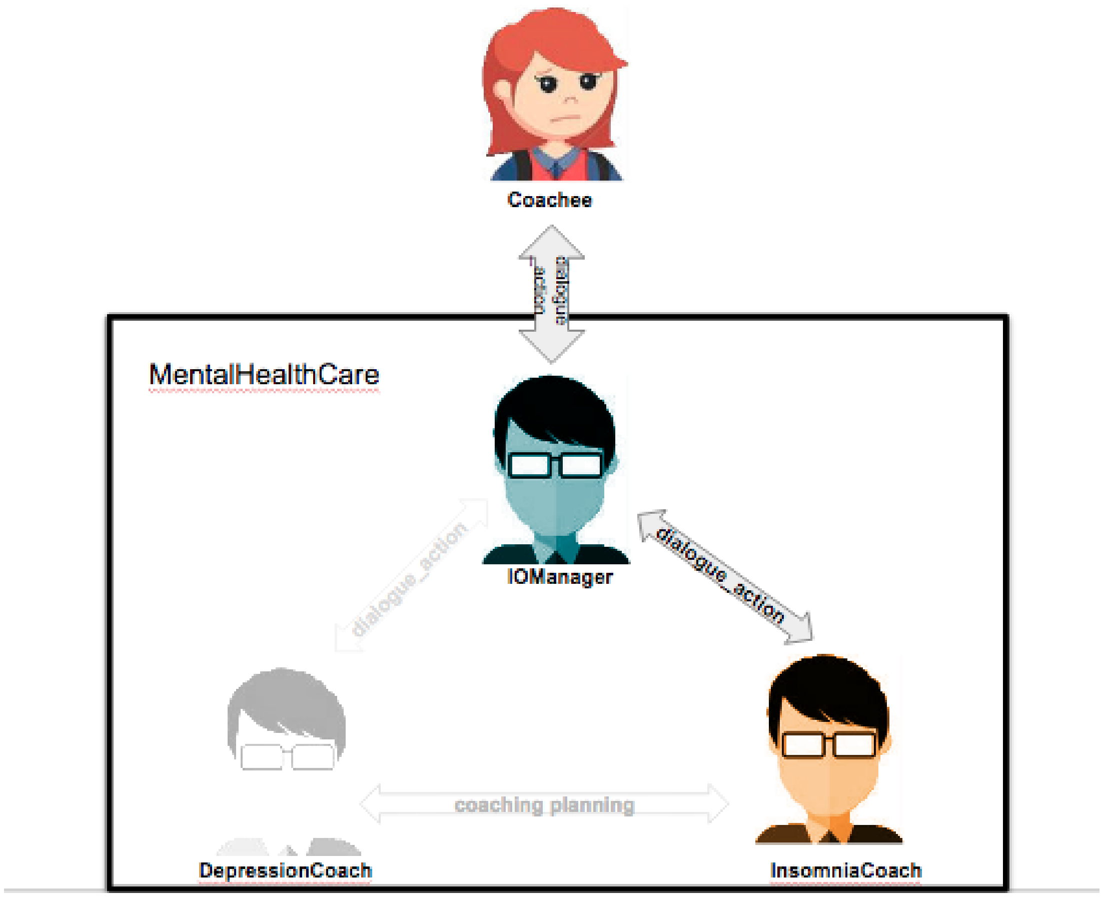

Figure 3.3: Interaction model of the insomnia coaching

direct interaction, that occurs via an Android device which may present itself to the coachee as certain tools like a sleep diary, or thought diary. Inside the system, the coaches receive or send information, or dialogues to the coachee via *IOManager* who inhabits in between the coaches and the coachee. The main purposes of having *IOManager* is to present the system as a single entity to the coachee, and regulate the *dialogue_actions* between the coaches and the coachee.

The coaches can communicate with each other through messages. Both coaches may use the same ACL to be semantically interoperable. The information that the exchanged messages contain may impact the knowledge-bases of the coaches through the *Updater* processes. The message format of different communication languages is almost similar; there are two significant parts of a message: performative and parameter [100]. From the point of view of OOP, a performative can be considered as a class; a parameter is an instance of a class, and a message is an object of that class. An example of a KQML message in MHC can be:
(tell

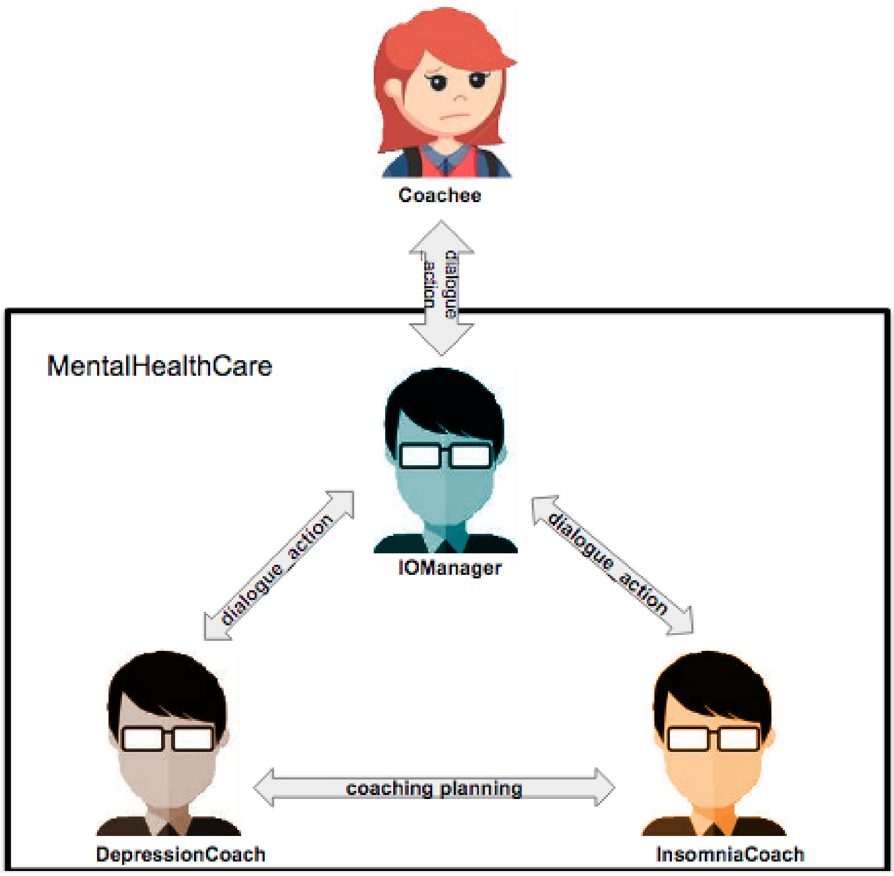

Figure 3.4: Interaction model of the combined coaching of depression and insomnia.

:sender DepressionCoach
:receiver InsomniaCoach
:ontology Schedule
:content start_time(Relaxation Exercise)
) [92]

The first line in the message is the performative which defines the illocutionary force (the sender's intention of producing this message). The last line is the content of the message that specifies the elementary form of a message. The other lines specify the sender, the receiver, and the ontology that holds the content. Through this message DepressionCoach is asking the InsomniaCoach to tell him about the start time of Relaxation exercise.

The coaches coordinate their actions through the *Schedule* tool which is common between both of them. However, this process is rather complicated, because coordinating plans and intentions may create complicated problems such as conflicts or disagreement between the coaches in regard to time, space, location, resource that they use in their plans. The primary reason behind such conflicts are differences in knowledge, expertise, inference mechanisms between the coaches. Conflict resolution is described in the next chapter.

3.6 Conclusion

Simply, In this chapter, the MAS paradigm with core concepts like what an agent is, what are the properties of the MAS, etc. and using the concepts we have modeled the internal architecture of a coach. We have also discussed how the MAS paradigm can be used for e-coaching. An agent can be viewed as an "Electronic Assistant" which assists its user by acting on behalf of him/her. According to Wooldridge and Jennings [73], an agent can be defined as a hardware or software-based computer system which holds the properties like i) autonomy: an agent is autonomous if it can operate without the direct interventions of humans or other components in the system; ii) social ability: an agent should be able to interact with other agents and humans via some communication mechanism; reactivity: an agent should have the ability to comprehend its environment and react to it in a timely fashion; proactivity: an agent should also be able to exhibit goal-directed behaviors by taking initiatives. Besides, the BDI-architecture characterizes an agent in terms of beliefs, desires, and intentions. A multi-agent system (MAS) can be defined as a group of entities who interact with each other to achieve individual or collective goal [77, 17]. In a MAS, the agents can automatically form a cooperating unity if there is an inter-action mechanism known as agent communication language (ACL) [93]. An ACL supports interoperability in a system if all the agents of the system ascribe the same meaning to the language, in other words, if they are semantically interoperable. On the top level, there are three components: a coachee, an e-coaching system, and an environment. Inside the e-coaching system, there are four agents: i) DepressionCoach, an expert in depression domain; ii) InsomniaCoach, an expert in insomnia domain; iii) IOManager, an input-output manager who manages the interactions between the coaches and the coachee; and iv) Scheduler who integrates the local plans of the coaches and schedule them on the Schedule after executing the conflict detection and resolution processes on them. A coach of MHC consists of three knowledge-bases, and various active processes. The main component of a coach is CoModel which contains knowledge bases for the coachee and is connected to the system clock to keep the model up to date. Tools like *local_schedule*, and *Contract*s, *Constraints* are part of the CoModel. Inside the CoModel, there is also a method called *checkConstraints*() which keeps a check on the coachee to see if she is following the *Constraint*s that she agreed to follow. If any conflict is detected, it sends it to the *Negotiator* process. The Negotiator generates suitable dialogue_actions using the *Inventory* and *Interaction_Recipe* to resolve the *Conflict*. And it communicates the dialogues to the coachee via the *IOManager*. In the next chapter, we will discuss conflicts in a MAS.

Chapter 4

Conflicts in MAS

In this chapter, we present literature review of conflicts in MAS. Section 4.1 gives a general definition of conflict; subsection 4.1.1 and subsection 4.1.2 discuss conflicts in humans and MASs. Section 4.2 provides the detail overview of various types of conflicts that can occur in a MAS. Section 4.3 describes the conflict detection process that can occur at design time or runtime. In section 4.4, we discuses the resolution processes that are used to resolve conflicts in MAS. Lastly, section 4.5 concludes the chapter.

4.1 What is Conflict?

In [7], conflict is defined as any situation of disagreement between two or more bodies, i.e., humans, software agents. Conflicts can occur in two major dimensions: i) internal conflict that occurs within one's mind (human) or knowledge-base (agent); and ii) external conflict that occurs between at least two humans or agents. In psychology, inner or internal conflict is perceived as a psychological struggle within one's mind which arises from opposing demands and impulses. For instance, often people experience conflicts between their desires [101]. In MASs, internal conflict may refer to conflicts that occur due to inconsistencies in an agent's knowledge-base or mental states, for instance, an agent has propositions p and $\neg p$ in his knowledge-base. In this thesis, we mainly discuss external conflicts that occur between two agents.

4.1.1 Conflict in Human

In human society, conflict refers to the situation or event when two or more persons or groups oppose each other with respect to their opinions or desires. For instance, in a company, conflicts can transpire between managers and subordinates or between employees and external stakeholders, i.e., customers. Often the conflicting parties exercise social power in an effort to achieve a scarce or incompatible goal and restrain the opponent to achieve it, e.g., using bribery to achieve the goal. Many conflict theorists like Karl Marx [102], Ludwig Gumplowicz [103], Lester F. Ward [104, 105] emphasize interests (self-interests) as the cause of conflicts in human society, for instance, the conflict between immigrants and native-born on the issue of immigration in many countries can be considered a conflict of interest.

In [106], three types of conflicts are distinguished: i) *conflict of appropriate interests*: two individuals interest the same thing and excluding one of them is the ultimate solution to the conflict (e.g., two persons applied for the same job and choosing or rejecting one of them is the solution to the conflict); ii) *inverse interests*: two individuals express opposing desires towards a particular thing (e.g., between two job interviewers, one likes an applicant while the other one does not like that applicant); iii) *incompatible interests*: two individuals desire two different things which are not compatible with each other (e.g., sales team and marketing team of a company adopt two different strategies to promote a new product, but the strategies are incompatible).

In [107], six types of conflict resolution model in human society are presented: a) *flight* (escaping from conflict), b) *destruction* (take over the opponent(s) to win the conflict), *subservience* (giving up), *delegation* (involving a third party to judge between the opponents), *compromise* (obtaining the result of negotiation of opponents), *consensus* (obtaining the result of agreement of opponents).

4.1.2 Conflict in MAS

From different research studies, i.e., [8, 76, 18], we can identify at least four distinct reasons that cause conflict between interacting agents in a MAS, such as:

i Decentralization is one of the key reasons for conflicts between agents [6]. In a decentralized MAS, agents have distinct domain knowledge. Consequently, they have incomplete knowledge about each other's knowledge domain which may cause conflict between them while they will cooperate or interact. For instance, in a self-driven car, the navigator agent (who controls the navigation) and the engine controller agent (who controls and manages different functions of the engine (i.e., ignition timing, fuel consumption)) may have incomplete or incompatible knowledge about each other's domains which may cause disagreements between them while driving, i.e., the navigator agent estimates that still 10 miles are remaining to reach the destination while the engine controller agent predicts that the car can go only 4 miles with the existing fuel.

ii When two or more agents collaborate to collectively accomplish a goal, there can be interdependency between them. For instance, the engine controller agent may depend on the navigator agent to calculate how many liters of fuel will be required to reach home. If navigator agent delays or denies due to his own pre-scheduled works or provides miscalculated information to the engine controller, there will be conflicts between them.

iii If there is scarcity of resources, or if the agents share common resources, there will be competition between them with respect to the resources, which will cause conflict in the system.

iv If the agents of a system have different mental states, i.e., beliefs, goals, plans or norms. While interacting with each other, they may get involved in conflicts due to the differences.

4.2 Classification of Conflicts

According to [76, 18], we can distinguish four main types of conflicts based on the mental notions of the agents, i.e., i) goal conflict, ii) plan conflict, iii) belief conflict, iv) normative conflict between two or more intelligent agents in regard to their mental states. The conflicts are described below.

i **Goal conflict:** Goal conflict arises when properties of two or more agents' goals do not comply with each other [76]. For example, for an automobile, optimization of speed and safety are two captivating goals. However, they cannot be achieved at the same time, and most of the time compromises are made to avoid such goal conflict.

ii **Plan Conflict:** Plan conflict involve properties of the agents' intended actions, i.e., precondition and postcondition where the precondition is a prerequisite that must be fulfilled for an action to be done, for instance, being conscious and not drunk is the precondition of driving a car. A postcondition is the effect of an action, for instance, accident is the postcondition of drinking and driving. If a precondition of an intended action becomes invalid due to the postcondition of its preceding action, conflict arises in the plan [76]. Plan conflicts generally arise when agents of a system integrate their personal or local plans to generate the global or final plan of actions to achieve the goals. For instance, the postcondition of having sedative at night might affect the next action of waking up early in the morning. Often reordering of plans resolves such conflicts. There are also plan refinement operators (promotion, demotion, separation, etc.) to resolve such conflicts.

iii **Belief Conflict:** Diversity in the agents' beliefs may also cause conflicts. Generally, beliefs involve propositions about facts and evaluations. If agents use different structures or abstraction levels to represent their beliefs, conflicts may transpire [76]. For instance, two agents are working on a project that involves outside temperature. The agents can sense the temperature, and both of them believe that it's quite high than normal. They plan their actions according to the current temperature and share their plans with each other. Despite, having similar beliefs and compatible action plans, conflicts arise because they use different units for temperature. The reason for the conflict can be either none of them is aware of each other's knowledge representation or each agent converts the other agent's unit to his own, and something goes wrong in this process.

iv **Normative Conflict:** In order to regulate and control an autonomous agent's behavior, norms, i.e., obligations, permissions, or prohibitions can be defined in the system's application domain. Conflicts between norms refer to the situations where two norms are categorically opposing and cannot be fulfilled together or one norm's fulfillment causes a violation of another one [18]. The two most common cases of normative conflicts arise between: 1) a norm of obligation and a norm of prohibition or 2) a norm of prohibition and a norm of permission.

We argue that all types of conflicts that may occur in a MAS can be classified into two groups: i) physical conflicts or resource conflicts, and ii) knowledge conflicts like Tessier has done in [107]. This classification is enough for addressing all types of conflicts that are produced between agents in a MAS because physical conflicts refer to conflicts between agents regarding their resource requirements, i.e., RAM space, whereas knowledge conflicts refer to propositional conflicts, and all the mental notions of individual agents can be represented as propositions, for instance, $Bel_a\, p$ is the propositional representation of agent a's belief which is stored in his knowledge-base.

i **Resource Conflicts:** Competing over limited resources is a key reason for conflicts. For instance, space is a critical resource for robot agents, and often resource conflicts occur when two or more robots want to be in the same space at the same time. Such conflicts can be resolved by either by expanding resources or by modifying one of the involved in conflict agent's (who are involved in the conflict) plan that requires the resource. Cr. Castelfranchi [108] describes resource or physical conflicts as extrinsic or nonanalytical conflicts.

ii **Knowledge Conflict:** Tessier believes that contradiction between agents' propositions is the mostly dealt with knowledge conflict [107]. Cr. Castelfranchi [108] also refers to goal or belief conflicts as knowledge conflict, perhaps because an agent's all the mental states including goals, beliefs are represented as propositions. Moreover, knowledge (dynamic knowledge) can be considered as the ground of all the mental states for a knowledge-based autonomous agent. For instance, $agent_a$ and $agent_b$ has the propositions $p \wedge q$ and $p \rightarrow \neg q$ respectively. The agents exchange their knowledge regarding p and q, and they realize their knowledge is not consistent.

All types of conflicts that arise in an application domain can be described as direct or indirect conflicts based on their detection process. Conflicts that can be detected by directly comparing the conflict subject (a set of matters on which the members are involved in a conflict) are categorized as direct conflicts. For example, the conflict between the beliefs of $agent_a$, $agent_b$ can be detected by directly comparing the conflict contents which are $Bel_a x$ and $Bel_b \neg x$. However, In indirect conflicts, conflict contents are not simple and can be composed of several related conflict subjects, i.e.,

4.3 Conflict Detection

Conflict detection is the triggering event for the conflict resolution, also crucial for coordination as it verifies whether the interactions between the agents are conflict-free [76]. Although different MASs may use different types of conflict detection process, the core of conflict detection is same which includes comparing the coaches' interactions or exchanged plans or information and looking for conflict among them. It may sound simple, but the detection process can be a very complicated and time-consuming for a distributed complicated multi-agent system with numerous agents as various types of conflicts, at different levels can arise in the system.

The conflict detection process can be performed at two levels: 1) *design time* which prevents the occurrences of conflicts during the execution of the MAS, or 2) *runtime* wherein the agents dynamically resolve the conflicts [18].

4.3.1 Conflicts Detection at Design time

At the design time, conflicts can be detected by comparing different components of the system ontologies of the agents. In [109], a "Conflict Checker" algorithm is presented which locally normalizes the relationships between the agents and their components (issues/matters about which the agents may have different perspectives or states), and find out if any two components overlap each other. For instance, A and B are two robots, and P is a place which can contain only one agent. A has the proposition $<A$ *inhabits* $P>$ in his ontology and "*inhabits*" is a typical relationship that relates an entity (here agent) to the environment (here place). The algorithm will look for the *inhabits* relationship in B's ontology. After finding the relationship, it will compare the related components. If B has $<B$ *inhabits* $Q>$ or $<B$ *inhabits* $\neg P>$, the algorithm concludes that there is no conflict between A and B regarding the place P. However, if B has $<B$ *inhabits* $P>$, the algorithm will conclude that there is a conflict between the two robots regarding the place P which is a resource conflict.

In [76], a local conflict representation model is used to detect conflicts at design time. The model represents the subjective perspectives, i.e., goals, plans, and beliefs of the agents living in an environment in different layers and the layers are connected with each other based on their relationships. The goal level represents the interdependencies between the goals and system resources. The plans levels represents the ordering relationships among the actions which are planned to achieve the goal. The belief level presents the supportive or attacking relationships between the beliefs and goals, and beliefs and plans. The same model can also be used for conflicts resolution as using the root of a conflict can be identified from the model. The model is refreshed periodically to keep it updated.

4.3.2 Conflicts Detection at Runtime

In [110, 111, 112, 113], a conflict detection model named NoA is presented that detects conflicts by comparing the agents' norms that govern their behaviors. It is assumed that as a part of coordination in a cooperative environment, the agents adopt each other's norms. For instance, agent A and B have norms $prohibition_x$ and $prohibition_y$. If A and B collaborate to accomplish $goal_z$, they will respect or adopt each others norms, if they do not there will be normative conflicts. However, before accepting any new norm an agent need to compare the new norm with the existing norms that he holds. Normative conflicts are detected by checking the intersection of sets of actions or states that partially or fully instantiated in the activity statements within norm specifications, i.e., in the above example, x, y are action statements in the norm statements. In addition, the indirect conflicts are detected by investigating all possible actions and compare side effects with the norm specifications. In order to avoid normative conflicts, the agents must investigate the selection and execution of actions. Nevertheless, the architecture generates permissive norms that allow agents to perform their plans, if there is no prohibition norms associated with the plans. In addition, the agents also maintain labeling of actions to classify and reason whether an action is norm compliant or not.

4.4 Conflicts Resolution

If any conflict is detected during the conflict detection process, it triggers the conflict resolution process which starts by looking for an optimal resolution to the conflict that is acceptable by everyone. The process also includes persuading others to accept the resolution if anyone decline it. Over the past two decades lots of research studies have been done to develop various conflict resolution strategies for MASs [114]. Below, we discuss the most popular conflict resolution strategies, i.e., *negotiation*, *arbitration*, *prioritizing*, *voting*, and *pay-off matrix*.

4.4.1 Negotiation

Negotiation is a well-known and accessible approach to conflicts resolution. In negotiation, compromises are made by one or more agents who are in conflict. The aim of negotiation is to find the compromise that is acceptable to all the agents. The agents of a system can participate in the negotiation process either directly (e.g., [18]) or via a mediator (e.g., [7]). In both cases, proposals of modifications or compromises are exchanged between the involved parties. The agents may also exchange feedbacks or evaluations about the trade-offs after evaluating them locally. If a mediator or planner is involved in the negotiation process, he also generates persuasive arguments to persuade the agents about his proposals of modifications. The architecture of mediator negotiation ensures two things: 1) the mediator has complete knowledge about the conflicts including the agents' action plans and states; 2) he may save the reactions of the agents to his proposals and their contexts to conflicts in order to reuse them in the future. If the agents agree to a solution, they make commitments to each other. Several types of commitments or deals can be used to resolve conflicts between the involved agents [8], i.e., 1) earliest start time: the committed agent agrees not to perform an action before the agreed time; 2) deadline: the agent agrees to accomplish his task before the agreed time; 3) do, the agent commits to perform the task in question; 4) don't: the agent agrees not to perform the task in question during the specified time period. The negotiation process involves both proactive and reactive behaviors of the agents. The agents need, and upon finding the optimal solution to the conflict, they need to offer commitments proactively. The involved agents also need to react to the proposals and offers after reasoning them so that they find an optimal outcome for themselves from the negotiation.

4.4.2 Arbitration

Arbitration is similar to the process of negotiation via a mediator or mediation. It is also conducted by a third part who may not have power to modify conflicting agents' behaviors. However, in arbitration, the decision of the third party or arbitrator must be accepted by the agents in conflict [114]. The arbitrator may have authority, more knowledge, and more solution search capabilities than other agents involved in conflict. In addition, the authority of the arbitrator comes from the agreement of agents in the organizational structure [115, 116].For instance, a civil court judge may play the role of an arbitrator for the people in dispute.

4.4.3 Prioritizing

Prioritization is a popular method that is used to resolve conflicts by ordering the components that are in a conflict based on their expected outcome or importance. Many aspects of the conflicting components like the situations they are in or their expected outcomes may play roles to determine the order of priority. In the priority allocation process, if a component gets a higher score, that means it will get higher priority and vice versa. In [117], prioritization model is used to control air traffic. In the airspace, conflict is defined as the event where two or more aircraft come too close; conflict detection is determining when a conflict can occur; conflict resolution is how a conflict can be avoided. The prioritization model maintains a priority list containing priority scores of the aircraft which may experience conflict. Criterions like distance to the destination, weather condition, speed level, altitude, distance from the other aircraft are used to assign them priority scores. Based on their priority values actions like change route or don't change route will be planned for the aircraft. When priority scores are assigned to each aircraft in conflict, the aircraft with the highest score will be prioritized to reach its destination without changing the route. Other aircraft will be asked to change their directions. Thus a potential conflict is resolved.

4.4.4 Voting

Voting is another popular approach to resolve conflicts. It is highly used in human society, for instance, government election. It is also effective for MASs to reach consensus and distribute authority unlike arbitration, i.e., [118] used voting mechanism to resolve air traffic control problems between aircrafts. Voting is an optimum approach to achieve maximum social welfare in a MAS where each agent expresses his preferences and votes for the component among all the components involved in a conflict that supports his goal and satisfies the system [119].

4.4.5 Self-Modification

In self-modification, an agent modifies his own behavior if he detects conflicts with other agents. It is also named as "Independence" [120]. It is a simple yet effective approach which does not require interaction among agents. For instance,

4.4.6 Pay-off Matrix

Using pay-off matrix is a game theoretic approach towards conflicts resolution. Rosenchein [121] first used this approach to resolve conflicts. The matrix contains pay-offs of each possible outcome of a strategic situation and the optimal solution for that situation is obtained the analysis of the matrix. For example, A & B are playing coin matching game where each of them has a coin. They will simultaneously show a side of their coin. All the possible outcomes of the situation hold certain pay-offs. If they choose head-head they will be rewarded by positive points. If one shows head while other one shows tail, the agent who shows tail will receive negative point as punishment. If both of them show tail, both of them will be punished. It is assumed that the agents will have prior knowledge of the pay-off matrix. This approach can be used for non-cooperative environments too.

4.5 Conclusion

In this chapter, we discuss what a conflict means in human society and agent society. We also discuss the different taxonomy of conflicts that occur between two agents of a MAS. Furthermore, we also explain different established conflict detection and resolution. Conflict is defined as any situation of disagreement between two or more bodies, i.e., humans, software agents. At least four issues have been identified from different research studies as the [19, 77, 18] that cause conflict between interacting agents in a MAS, such as, i) decentralization, ii) task interdependencies iii) resource scarcity or shared resource, iv) differences in mental states, i,e., belief, goal, plan, norm. Due to the difference in mental notions, two agents can have at least four types of conflicts [77, 18], i.e., i) goal conflict, ii) plan conflict, iii) belief conflict, iv) normative conflict. However, it is argued that there are only two major types of conflicts that occur in a MAS: i) physical conflicts or resource conflicts, and ii) knowledge conflicts [108]. This classification is enough for addressing all types of conflicts that are produced between agents in a MAS because physical conflicts refer to conflicts between agents regarding their resource requirements, i.e., RAM space, whereas knowledge conflicts refer to propositional conflicts, and all the mental notions of individual agents can be represented as propositions. Conflict detection is the triggering event for the conflict resolution, also crucial for coordination as it verifies whether the interactions between the agents are conflict-free [77]. The detection process can be performed at two levels: 1) design time which prevents the occurrences of conflicts during the execution of the MAS, or 2) runtime. If any conflict is detected during the conflict detection process, it triggers the conflict resolution process which starts by looking for an optimal resolution to the conflict that is acceptable by everyone. Among all resolution techniques negotiation, arbitration, prioritization, and self-modification. Negotiation is a well-known and accessible approach to conflicts resolution. In negotiation, compromises are made by one or more agents who are in conflict. The aim of negotiation is to find the compromise that is acceptable to all the agents. The agents of a system can participate in the negotiation process either directly (e.g., [18]) or via a mediator (e.g., [16]). Arbitration is similar to negotiation via a mediator or mediation. It is also conducted by a third party who may not have the power to modify conflicting agentsâĂŹ behaviors. Prioritization is a popular method that is used to resolve conflicts by ordering the components that are in a conflict based on their expected outcome or importance. Many aspects of the conflicting components like the situations they are in or their expected outcomes may play roles to determine the order of priority. In self-modification, an agent modifies his own behavior if he detects conflicts with other agents. t is a simple yet effective approach which does not require interaction among agents.

Chapter 5

Theory into Practice: Integrating the Two Therapies of E-coaching

In this chapter, we discuss how the two different therapies are embodied and integrated in the e-coaching system. In section 5.1, we explain how the therapeutic techniques or exercises are supported by various built-in tools. The integration of two therapies are discussed in section 5.2. In reference to the integration process, the similarities and differences between the two therapy domains are explained in subsection 5.2.1 and subsection 5.2.2 respectively. Section 5.3 addresses the conflicts that can occur between the coaches after the integration. Lastly, section 5.4 concludes the chapter. Moreover, the ontologies of CBT-I and CBT-D are given in Appendix A and Appendix B for more details.

5.1 Integration of Coaching Tools

The coaching process incorporates two types of actions: i) *main actions* that must be performed and experienced by the coachee, for example, sleep restriction, thoughts recording; and ii) *supporting actions* that are designed to help the coachee in the coaching process such as scheduling exercises for the coachee or examining her performances in different exercises and produce information from them, i.e., calculating sleep efficiency. The e-coaching system supports different types of therapeutic techniques or exercises through built-in tools or tools which help the coachee to accomplish the exercises, i.e., a paper and pencil sleep diary of CBT-I or thought diary of CBT-D are replaced by electronic versions in MHC which supports exercises like recording sleep times or thoughts. The tools are also significant because they automatically collect valuable information from the coachee to personalize the intervention. For instance, the e-coaching system can automatically identify the patterns of automatic thoughts in the thought diary, and the e-coach can consult about the particular thoughts with the coachee.

The tools are modeled as interactive user interfaces that give the coachee access to the domain of discourse (observation and/or manipulation) and assist her by relevant information, guidance, and resources (i.e., electronic diaries to store, compare and overview data) to perform therapeutic exercises. Each tool is designed for a particular purpose or use. According to their uses, they may contain linguistic (i.e., words, sentences, and even paragraphs), visual (i.e., video, picture) and auditory elements (sound). Below different cognitive and behavioral tools of MHC that support CBT-I and CBT-D exercises are explained. The tools are designed by [33].

- **SleepDiary**: For the insomnia therapy, SleepDiary is the tool to store sleep relevant data, i.e., times of going to bed and waking up from sleep, sleep quality.

- **ActivityDiary**: It is the tool where the coachee inserts her daily life activities with name, time, rating on the basis of pleasure, and activated automatic thoughts during performing that activity.

- **ThoughtDiary**: It is an efficient tool which serves for several cognitive techniques, i.e., identifying automatic thoughts, investigation of the automatic thoughts, reattribution, and alternative solutions of those thoughts. This tool may consist of 7 interfaces, i.e.,

  I The first page is *Thoughts* where the coachee registers her thoughts. It helps the coachee to distinguish an automatic thought from her regular thoughts, i.e., if a particular type of thought appears more often regardless of the type of event or situation, and it affects the coachee negatively, it is considered as an automatic thought. For instance, the coachee feels "she is worthless" during many regular situations like classes in school, movie outing with friends, and eating out with parents which clearly shows that this is an automatic thought.

  II The second page is *Situation* where the coachee registers the situation or event that has stimulated the thought on the first page. In the previous example, classes in school, movie outing with friends, and eating out with parents are all situations that activated the automatic thought.

  III Third page is *Hypothesis* where she registers the potential reason/rationale for having that thought in that situation, i.e., in the above example, "she did not want to be in those situations" is the hypothesis of feeling worthless in those situations.

  IV Fourth page is *Evidence* where she notes down the evidences in support of her hypothesis on the previous page, i.e., her mom or friends compelled her to be in those situations.

V Fifth page is *Retest* which compels the coachee to ask herself one more time whether the evidences are true and do they really prove her hypothesis.

VI Sixth page is *Alternative Solution* for finding some alternative thoughts which can be logically related to that particular situation, i.e., instead of feeling worthless, the coachee could have felt low during the class because she did not understand the lectures.

VII Seventh page is *Judgement* where the coachee judges her thoughts against all the information provided in previous pages.

- **Relaxation**: It is used in sleep therapy and offers a spoken muscle relaxation exercise of various length (1, 2, 4, 8, and 16 min).
- **Distraction**: The tool suggests several relaxing activities that can distract the coachee from thinking negative things, i.e., physical activities or music.
- **Self-help**: Besides, these tools, there can be some general purpose tools also which may contain general information of both therapies. For instance,There can be tools to provide non-personalized information and guidance for each type of therapy and technique in the text, audio, picture, or video format.

One of the important tools for the coaches to plan and monitor the coaching process is *Schedule*. This tool is used to schedule the coaching process from the beginning till the end. It is basically the integrated product of the local schedules or local_schedules of the coaches. A coach locally schedules the exercises for the coachee on his local_schedule and then share it with the Scheduler agent to schedule them on the Schedule. Schedule also gives access to the coachee to the coaching process. We will learn more about this integration process in the next chapter.

5.2 Integration of CBT-I and CBT-D

We already know that MHC consists of two intelligent expert agents or coaches, i.e., *InsomniaCoach* and *DepressionCoach* who have expertise on CBT-I and CBT-D respectively by having independent knowledge-bases. Moreover, they can plan the coaching on insomnia or depression independently by using their knowledge-bases. Therefore, to coach a coachee who has symptoms from both depression and insomnia disorders, the coaches need to cooperate with each other and thus integrate their knowledge and plans for the coaching process. In MHC, cooperation between the coaches and/or integration of CBT-I and CBT-D are modeled in three steps:

I **Distribution of symptoms:** In the opening phase of the e-coaching, after diagnosing the symptoms of the coachee, the coaches distribute the symptoms between them according to their knowledge. The symptoms of insomnia include: difficulty in falling asleep, difficulty in maintaining sleep, early morning awakening and trouble in going back to sleep (for at least 3 nights per week and thus for at least 3 months); tiredness, stress, and irritation in the daytime due to lack of sleep [122]. And, the symptoms of depression include: loss of interest or pleasure, depressed mood, insomnia or hypersomnia, poor appetite or overeating, feelings of worthlessness or guilt, lack of concentration, psychomotor agitation or retardation, and suicidality [123]. So if a coachee has symptoms of low mood, overeating, and trouble falling asleep, DerepssionCoach will plan therapy for low mood and overeating whereas InsomniaCoach will plan therapy for trouble falling asleep.

II **Integration of local schedules:** In the plan integration phase, the coaches share their *local_schedule*s with each other to integrate the exercises on the Schedule and share them with the coachee. If there is no conflict between the exercises of different *local_schedule*s, the exercises are entered on the *Schedule* tool. For instance, DepressionCoach plans activity scheduling and thoughts recording exercises on his *local_schedule*. And, InsomniaCoach plans filling in sleep diary and stimulus control exercises for the coachee on his local_schedule. In the plan integration phase, they share their *local_schedule*s with each other to outline the final coaching plan for the coachee combining exercises from both domains.

III **Conflict resolution:** However, the coaches may unintentionally or unconsciously conflict with each other through their *local_schedule*s. For instance, the coaches schedule activity scheduling and filling in sleep diary at the same time. However, the exercises require distinct tools to be done, i.e., ActivityDiary and SleepDiary which are two different interfaces of the application that cannot be accessed simultaneously. Thus, the coaches are in conflict with each other which may prevent them from operating the coaching program successfully. After integrating the local schedules, the coaches look for such conflicts, and if they find any, they resolve them cooperatively.

5.2.1 Similarities between CBT-I and CBT-D

The therapies of insomnia and depression are similar in many ways like diagnostic criteria or symptoms, patterns of therapeutic exercises, and interaction recipes (persuasive strategies). There are many symptoms which are common in both disorders, for instance, sleep disturbance, tiredness or fatigue, irritation, and distress. CBT-I and CBT-D are also similar in the therapeutic techniques they use. For instance, both therapies emphasize scheduling exercises, i.e., activity scheduling of CBT-D, and filling in sleep diary of CBT-I; exercises of recording daily information, i.e., thoughts recording of CBT-D, filling in sleep diary of CBT-I; exercises for relaxation of mind and body, i.e., distraction exercise of CBT-D, relaxation exercise of CBT-I; and interactive exercises, i.e., cognitive rehearsal, self-reliance, role playing, role reversal of CBT-D and briefing of CBT-I.

We have seen in the previous chapter, that an The therapeutic exercises of both therapies are also similar in how they are defined in the coaches' knowledge-bases and represented to the coachee. An exercise contains three types of primary attributes, i.e., time, proposition, and resources. Time-attributes include the start time and duration of an exercise; propositional attributes include the precondition and postcondition of an exercise; resource- attributes include the tools an exercise requires to be performed. Besides, there is another attribute, i.e., planner that defines whether an exercise belongs to CBT-I or CBT-D. This attribute is essential to detect conflicts between the coaches. The attributes to define and represent an exercise are defined below:

I **Time-attributes:** the time-attributes include starting time and ending time of an Exercise. We define a time point as hour:minute:seconds formats.

II **Propositional-attribute:** Preconditions and postconditions are perhaps the most important criteria to define an exercise. A precondition is a prerequisite that must be fulfilled for an exercise to be performed, for instance, quietness is the precondition of the relaxation exercise. A postcondition is the effect of an action, for instance, a relaxed body and mind are the postconditions of the relaxation exercise. Also, the precondition and postcondition of an exercise are expressed as propositions. Generally, preconditions and postconditions refer to particular situations like *inBed*, *atHome*, *inBedRoom*, *lightDim*, *quiet*, *awake*, *awake*, *eat*, *play*, *work*, *sleep*, *sleepEfficiency* > 10, *identifyAutomaticThoughts.* Preconditions and postconditions can also consist of multiple propositions, for instance, the precondition of the relaxation exercise is $\neg$ eating, $\neg$ sleeping, $\neg$ atHome. In such case, the precondition or postcondition will be expressed as a set of propositions, i.e., $\{\neg$ eating, $\neg$ sleeping, $\neg$ atHome$\}$. If all of the propositions are true, only then the precondition is true. In other words, we can also say that a precondition or postcondition is a set of propositions where the propositions have only two values: true or false. And if all the propositions are true, then the precondition or the postcondition is true.

III **Resource-attribute:** The resource-attribute describes the resources that an exercise needs to be performed. For instance, a tool is considered a resource for the exercise to be performed.

IV Planner: The planner attribute guides the Scheduler to understand which exercise is planned which coach. For instance, relaxation.planner = InsomniaCoach, but thought_recording.planne = DepressionCoach.

CBT-I and CBT-D also show similarities in their use of tools and resources to perform exercises. For instance, a) the same interactive tool could be used to perform many exercises; b) both coaches share *Schedule* tool (the coachee also has access to the tools, especially Schedule); c) physical resources like actor of the exercise (the coachee is the common actor of all the exercises), sensors (i.e., wearables), location where a particular exercise is supposed to be performed, and time when a particular exercise is supposed to be performed.

5.2.2 Differences between CBT-I and CBT-D

CBT-I and CBT-D have many intervention techniques which are distinct in their approaches and methods, for instance, techniques of CBT-I are concentrated in regulating sleep time and efficiency whereas techniques of CBT-D have broader purposes, like disciplining the coachee's daily life routine, correcting her thinking process. Not only CBT-D has more exercises than CBT-I, but it also incorporates more interactive exercises between the e-coach and the coachee, for instance, cognitive rehearsal, self-reliance, role playing, role reversal, briefing, reattribution, alternative solutions which may take longer time than the exercises of CBT-I. The treatment of insomnia may take 6-10 weeks of time. However, it cannot be determined how much time the treatment of depression may take beforehand.

5.3 Conflicts between CBT-I and CBT-D

The coaches may conflict each other with respect to their knowledge and limited resources. Conflicts between them can arise from the beginning of the coaching such as in the distribution of the symptoms phase. In this phase, the coaches may conflict because both of them have expertise on the same symptoms, i.e., sleep disturbance, fatigue, and their way of dealing the symptoms might be different. In the integration of the local schedules, the coaches may conflict concerning the properties of the exercises such as:

I The actor of the exercises (who is supposed to perform the exercises, i.e., the coachee) may not be able to perform two or more distinct exercises at the same time. For instance, the coachee may not perform the relaxation exercise of CBT-I and the role-playing exercise of CBT-D simultaneously. If the coaches schedule exercises in a way that the coachee has to perform two or more exercise that she cannot perform simultaneously, the coaches get in conflict.

II The place where the exercises are supposed to be performed. If two or more exercises that are scheduled to be performed simultaneously by the same actor or coachee but in different places, the coaches are in conflict in regard to the places of the exercises.

III The coaches may also get involved in conflicts regarding the relation between the preconditions and postconditions of the exercises. The planned exercises may affect each other negatively if the preconditions of an exercise get invalidated by the postconditions of the former exercises. In other words, if the result of an exercise prevents its following exercise to be performed by invalidating its preconditions. Such situations are also considered as conflicts. For instance, doing briefing exercise in the bedroom violates the precondition of the stimulus control exercise.

IV Desire to use the same tool for two different simultaneous exercises also causes conflict between the coaches. For instance, if briefing and cognitive rehearsal or role-playing are scheduled at the same time, it will cause conflict as the exercises are done using Talk tool.

The coaches may also conflict with each other in the conflict resolution phase regarding choosing the potential solution to a conflict, i.e., they may not agree on the same solution.

5.4 Conclusion

In this chapter, we have discussed how CBT-I and CBT-D can be integrated in MHC and what kind of conflicts may arise due to this integration. The coaching process incorporates two types of actions: i) *main actions* and ii) *supporting actions*. Main actions are the exercises that must be performed and experienced by the coachee, for example, sleep restriction, thoughts recording and the supporting actions are designed to help the coachee in the coaching process such as scheduling exercises for the coachee or examining her performances in different exercises. The e-coaching system supports different types of therapeutic techniques or exercises through built-in tools which help the coachee to accomplish the exercises, i.e., sleep diary of CBT-I or thought diary of CBT-D support exercises like recording sleep times or thoughts. The tools are also significant because they automatically collect valuable information from the coachee to personalize the intervention. In MHC, cooperation between the coaches and/or integration of CBT-I and CBT-D are modeled in three levels: distribution of symptoms plans integration, and conflict resolutions. In the plan integration phase, the coaches share their *local_schedule*s with each other to integrate the exercises on the Schedule and share them with the coachee. If there is no conflict between the exercises of different *local_schedule*s, the exercises are entered on the *Schedule* tool. An exercise contains four types of primary attributes, i.e., time, proposition, resources, and planner. Time-attributes include the start time and duration of an exercise; propositional attributes include the precondition and postcondition of an exercise; resource- attributes include the tools an exercise requires to be performed. Besides, there is another attribute, i.e., planner that defines whether an exercise belongs to CBT-I or CBT-D. This attribute is essential to detect conflicts between the coaches. CBT-I and CBT-D have many intervention techniques which are distinct in their approaches and methods, for instance, techniques of CBT-I are concentrated in regulating sleep time and efficiency whereas techniques of CBT-D have broader purposes, like disciplining the coachee's daily life routine, correcting her thinking process. The coaches may conflict each other with respect to their knowledge and limited resources. Conflicts between them can arise from the beginning of the coaching such as in the distribution of the symptoms phase if both of the coaches have expertise on the same symptoms. The coaches may also get involved in conflicts regarding the relation between the preconditions and postconditions of the exercises. The planned exercises may affect each other negatively if the preconditions of an exercise get invalidated by the postconditions of the former exercises. Desire to use the same tool for two different simultaneous exercises also causes conflict between the coaches. The coaches may also conflict with each other in the conflict resolution phase regarding choosing the potential solution to a conflict, i.e., they may not agree on the same solution. In the next chapter, we will discuss the conflict in MHC in more details and will model the conflict detection and resolution mechanism to handle the conflicts in MHC.

Chapter 6

Conflicts Resolution in MHC

In this chapter, we present the conflict resolution model for MHC that can detect and resolve conflicts between the coaches. In section 6.1, we classify the conflicts that may occur between the coaches in MHC. We provide the formal definitions of the conflict in section 6.2. In section 6.3, we present *Scheduler* as the conflict resolution model for MHC. In subsection 6.3.1, we discuss the basic scheduling process that the *Scheduler* uses to schedule exercises on the *Schedule*. And in subsection 6.3.2, we explain how the *Scheduler* schedules exercises which are in conflicts. In section 6.4 and section 6.5, we describe the resolution techniques that the *Scheduler* uses to resolve the conflicts. In example 6.5.1, we show how the *Scheduler* works with an example. Lastly section 6.6 concludes the chapter.

6.1 Conflicts in MHC

Conflicts between the coaches may occur at two different levels, i.e., i) distribution of the symptoms, ii) integration of the local schedules. Conflicts regarding the distribution of the symptoms can be resolved by implying certain norms in the e-coaching system, i.e., if there is a particular symptom that is common between the two domains, the coach who holds the greater knowledge about that symptom should plan exercises to treat it. And such a norm can resolve conflicts between InsomniaCoach and DepressionCoach regarding the common symptoms, i.e., "sleep disturbance" or "fatigue," i.e., InsomniaCoach has more knowledge about "sleep disturbance," so between the two coaches, he should be the one to plan exercises for this symptoms. Another resolution technique could be distributing symptoms based on their root causes, for instance, "fatigue due to low mood" is DepressionCoach's expertise, in contrast, "fatigue due to lack of sleep" is InsomniaCoach's expertise.

In the integration phase, there can arise many conflicts between the coaches, i.e., conflicts regarding actor of the exercise, location of the exercise, relationships between the exercises, and tools required by the exercises. However, We identify two main types of conflicts that cover all the conflicts that may occur in the plan integration phase:

I **Resource Conflicts:** This group of conflicts incorporate conflicts that arise due to overlaps in resource use such as actor, location, tool or time.

II **Knowledge Conflicts:** Knowledge conflicts involve more complex attributes such as the representation of knowledge or propositions that play a crucial role in planning the coaching process. In fact, preconditions and postconditions of the exercises are represented as propositions in the coaches' knowledge bases. Hence conflicts regarding these attributes are referred to as knowledge conflicts.

In the subsequent sections, we define the two types of conflicts with their formal definitions.

6.1.1 Knowledge conflict

Knowledge conflicts may occur between the coaches because of their independent knowledge-bases which contain their views, perceptions, and information about the coachee and her environment. Knowledge conflicts particularly refer to the situation of the plan integration phase where an exercise from a coach's *local_schedule* negatively affects an exercise from the other coach's *local_schedule*. And an exercise can affect another exercise when the former exercise's postcondition invalidates the precondition of the later exercise.

For instance, physical exercise (distraction exercise) and briefing are two consecutive exercises on the *Schedule* where physical exercise belongs to CBT-D and planned by the DepressionCoach, and briefing exercise belongs to CBT-I and planned by the InsomniaCoach. The postcondition of the physical exercise include { ¬at_home, in_gym} whereas the precondition of the briefing exercise include { at_home, ¬ sleeping, ¬ eating}. If the situation does not change after the execution of physical exercise and before the execution of the briefing exercise, then after doing physical exercise the coachee is not at home which is preventing her from doing the briefing exercise because it requires the coachee to be at home.

6.1.2 Resource Conflict

We identify that if exercises overlap in time, they may also overlap in other resource uses like tools, location, and actor. Hence, we discourage time overlap of the exercises as we believe that two exercises should not overlap because doing multiple exercises at the same time divides the coachee's attention between the exercises which may not produce the desired outcome. For instance, if thought recording exercise and relaxation exercise are done simultaneously, the coachee may not be able to focus on a single exercise. Hence, if two exercises overlap in time, we address it as resource conflict.

6.2 Formal Definitions

Let's have a look at all the syntaxes that we use to define all the attributes:

- An exercise is denoted by α. And $\forall \alpha, \alpha = \{p, t_s, l, pre, post, to\}$ such that:
- t_s is the start time of the exercise. For two exercises α_1 and α_2, if $\alpha_1.t_S < \alpha_2.t_s$ that means α_1 occurs earlier than α_2. Similarly, if $\alpha_1.t_S > \alpha_2.t_s$ that means α_1 occurs after α_2. And if $\alpha_1.t_S = \alpha_2.t_s$ that means α_1 and α_2 occur simultaneously.
- l is the length or duration of the exercise. For an exercise α that has start time t_s and duration l, the end time of α is $\alpha.t_s + \alpha.l$ or $\alpha.(t_s + l)$.
- pre denotes the preconditions of the exercise.
- $post$ denotes the postconditions of the exercise.
- to denotes the tool or list of tools that a coachee is offered to perform the exercise.
- p denotes the planner of the exercise. If $\alpha.p = d$, the exercise is planned by the DepressionCoach and such an exercise is referred to as α_d . And if $\alpha.p = i$, the exercise is planned by the InsomniaCoach and the exercise is referred to as α_i.
- s_d and s_i are the two local schedules of DepressionCoach and Insomnia-Coach. s_d contains the exercises that are planned by the DepressionCoach. So, $s_d = \{\alpha_{d_k} | 0 < k \leq N\}$ and $s_i = \{\alpha_{i_k} | 0 < k \leq N\}$. k,N are natural numbers.
- $\rightarrow\leftarrow$ denotes conflicts between two components, i.e., $\alpha_i \rightarrow\leftarrow \alpha_d$ means that the exercises α_i and α_d are in conflict, or $DepressionCoach \rightarrow\leftarrow InsomniaCoach$ means that the coaches DepressionCoach and Insomnia-Coach are in conflict.

Resource Conflict

Two exercises α_x and α_y which are not planned by the same coach, i.e., $\alpha_x.p \neq \alpha_y.p$ may conflict with each other:

- if the start times of the exercises are equal, i.e., $\alpha_x.t_s = \alpha_y.t_s$;

- or if their start times are not equal to each other and let's say α_x starts earlier than α_y i.e., $\alpha_x.t_s < \alpha_y.t_s$, but α_y starts before α_x could finish or the end time of α_x is not less than or equal to the start time of α_y, i.e., $\alpha_x.(t_s + l) \nleq \alpha_y.t_s$)

$$\forall \alpha_x, \alpha_y; \alpha_x \rightarrow\leftarrow \alpha_y \, if \, \alpha_x.p \neq \alpha_y.p$$
$$\wedge((\alpha_x.t_s = \alpha_y.t_s) \vee (\alpha_x.t_s < \alpha_y.t_s \wedge \alpha_x.(t_s + l) \nleq \alpha_y.t_s))$$

Knowledge Conflicts

Knowledge conflict occurs when between two exercises which are planned by the different coaches and do not have resource conflict, the former exercise's postcondition holds the negation of its following exercise's precondition.

$$\forall \alpha \exists \alpha_x, \alpha_y; \alpha_x \rightarrow\leftarrow \alpha_y \, if \, \alpha_x.p \neq \alpha_y.p \wedge ((\alpha_x.t_s < \alpha_y.t_s) \wedge \alpha_x.(t_s + l) \nleq \alpha_y.t_s))$$
$$\wedge \exists p(p \in \alpha_y.pre \wedge \neg pin \alpha_x.post)$$

6.3 Scheduler: A Model for Scheduling and Conflict Resolution

In this section, we introduce the fourth agent of MHC named *Scheduler* who makes the final call in scheduling exercises for the coachee on the *Schedule*. He is in charge of sharing the exercises that are planned in the *local_schedule*s by the coaches with the coachee via the Schedule tool. After preparing exercises, the coaches send their *local_schedule*s to the *Scheduler* to insert them on the *Schedule*. The Scheduler assigns the exercises on the time slots of the *Schedule* as per their start times and duration. He implements *the activity schedule mechanism* which includes scheduling exercises on the *Schedule*, and keeps track of various exercises that have been done or should be done during the coaching process.

6.3.1 Basic Scheduling Process without Conflicts

We develop the basic scheduling algorithm, i.e., algorithm 1 that cannot detect or resolve conflict, but only schedules the exercises according to their start times on the *Schedule*. The algorithm is built on the ground of two critical assumptions: i) all the exercises of the *local_schedule*s are consistent with each other and there is no probability of having conflicts between the coaches; ii) In a *local_schedule*, exercises are sorted according to their start times.

The Scheduler schedules or enters exercises on the Schedule according to their start times. And he starts off with the exercise that has lowest start time. As per the second assumption, the first exercise of a *local_schedule* has the lowest

Algorithm 1 Basic scheduling algorithm

```
 1: procedure SCHEDULER($s_d, s_i$)
 2:     while $length(s_i) > 0$ and $length(s_d) > 0$ do
 3:         if $\alpha_{i_1}.t_s < \alpha_{d_1}.t_s$ then
 4:             insert_Schedule($\alpha_{i_1}$)          ▷ insert the $\alpha_{i_1}$ on the Schedule
 5:             remove($s_i, \alpha_{i_1}$)                    ▷ remove $\alpha_i$ from $s_i$
 6:         else
 7:             insert_Schedule($\alpha_{d_1}$)
 8:             remove($s_d, \alpha_{d_1}$)
 9:         end if
10:     end while
11:     if $length(s_i) = 0$ and $length(s_d) > 0$ then
12:         for each $\alpha_d$ in $s_d$ do
13:             insert_Schedule($\alpha_d$)
14:             remove($s_d, \alpha_d$)
15:         end for
16:     else if $length(s_i) > 0$ and $length(s_d) = 0$ then
17:         for each $\alpha_i$ in $s_i$ do
18:             insert_Schedule($\alpha_i$)
19:             remove($s_i, \alpha_i$)
20:         end for
21:         if $length(s_i) = 0$ and $length(s_d) = 0$ then
22:             send_message("All exercises are scheduled.")
23:         end if
24:     end if
25: return Schedule
26: end procedure
```

start time. If there are exercises on both *local_schedule*s, the Scheduler starts the scheduling process by comparing the start times of the first exercises of the two *local_schedule*. The exercise with the lower start time is inserted on the *Schedule*, and removed from its *local_schedule*. This process will continue, until one or both of the *local_schedule*s become empty. If there is no exercise in one of the *local_schedule*s, the Scheduler will schedule the exercises of the *local_schedule* as they are in the *local_schedule* without any checking. If both of the local schedules do not have any exercise left to be scheduled, the *Scheduler* sends the message to the coaches that it scheduled all the exercises and no exercise is left to be scheduled.

Syntaxes

- s_i and s_d refer to InsomniaCoach and DepressionCoach's local_schedules.

- length(*local_schedule*) provides the number of exercises that *local_schedule* holds that are not scheduled on the Schedule. length(local_schedule) is always greater or equal to 0.
- α_{i_1}, α_{d_1} are the exercises with the lowest start times in s_i and s_d. The subscript refers to the index of the exercises in a local_schedule.
- "<" denotes the precedence of the exercises, i.e., $\alpha_1 < \alpha_2$ denotes that α_1 is the former exercise which has start time lower than α_2.

Example 6.3.1. Let's assume, s_d={thought_recording, cognitive_rehearsal} where thought_recording.t_s = 20:00, cognitive_rehearsal.t_s = 21:00. And, s_i={relaxation, sleep_hygiene, sleep_restriction} where relaxation.t_s = 20:30, sleep_hygiene.t_s = 21:30, sleep_restriction.t_s = 22:00 (see tables 6.1, 6.2). And there is no exercise scheduled on the Schedule at this point (6.3)

s_d	
thought_recording	cognitive_rehearsal

Table 6.1: s_d at the beginning of the Scheduling process.

s_i		
relaxation	sleep_hygine	sleep_restriction

Table 6.2: s_i at the beginning of the Scheduling process.

Schedule

Table 6.3: Schedule at the beginning of the Scheduling process.

- Step 1. length(s_d) = 2 and length(s_i) = 3 which satisfies the while loop at line 2. α_{i_1} = relaxation and α_{d_1} = thought_recording. And $\alpha_{i_1}.t_s$ = 20:30 and $\alpha_{d_1}.t_s$ = 20:00. $\alpha_{i_1}.t_s \nleq \alpha_{d_1}.t_s$. Hence, the process enters into the else condition at line 6. The thought_recording exercise is inserted on the Schedule by the method insert_Schedule() at line 7, and gets removed from s_d at line 8 (see tables 6.4, 6.5, 6.6).
- Step 2, length(s_d) = 1 and length(s_i) = 3. So the process enters into the while loop at line 2. α_{i_1} = relaxation and α_{d_1} = cognitive_rehearsal. And $\alpha_{i_1}.t_s$ = 20:30 and $\alpha_{d_1}.t_s$ = 21:00. $\alpha_{i_1}.t_s \leq \alpha_{d_1}.t_s$. Hence, the process enters into the if condition at line 3. The relaxation exercise is inserted on the Schedule by the method insert_Schedule() at line 4, and gets removed from s_i at line 5 (see tables 6.7, 6.8, 6.9).

s_d
cognitive_rehearsal

Table 6.4: s_d after step 1.

s_i		
relaxation	sleep_hygine	sleep_restriction

Table 6.5: s_i after step 1.

Schedule
thought_recording

Table 6.6: *Schedule* after step 1.

s_d
cognitive_rehearsal

Table 6.7: s_d after step 2.

s_i	
sleep_hygine	sleep_restriction

Table 6.8: s_i after step 2.

Schedule	
thought_recording	relaxation

Table 6.9: *Schedule* after step 2.

- Step 3, length(s_d) = 1 and length(s_i) = 2. So the process enters into the while loop at line 2. α_{i_1} = sleep_hygine and α_{d_1} = cognitive_rehearsal. And $\alpha_{i_1}.t_s$ = 21:30 and $\alpha_{d_1}.t_s$ = 21:00. $\alpha_{i_1}.t_s \nleq \alpha_{d_1}.t_s$. Hence, the process enters into the else condition at line 6. The cognitive_rehearsal exercise is inserted on the Schedule by the method insert_Schedule() at line 7, and gets removed from s_d at line 8 (see tables 6.10, 6.11, 6.12).

s_d

Table 6.10: s_d after step 3.

- Step 4, length(s_d) = 0 and length(s_i) = 1 which satisfies the else if condition at 16. So the process enters into the for loop at line 17.

s_i	
sleep_hygine	sleep_restriction

Table 6.11: s_i after step 3.

Schedule		
thought_recording	relaxation	cognitive_rehearsal

Table 6.12: *Schedule* after step 3.

- iteration 1, sleep_hygine is inserted on the Schedule at line 18 and removed from s_i at line 19 (see tables 6.13, 6.14).

s_i
sleep_restriction

Table 6.13: s_i after step 4, iteration 1.

Schedule			
thought_recording	relaxation	cognitive_rehearsal	sleep_hygine

Table 6.14: *Schedule* after step 4 iteration 1.

- iteration 2, sleep_restriction is inserted on the Schedule at line 18 and removed from s_i at line 19 (see tables 6.15, 6.16).

s_i

Table 6.15: s_i after step 4 iteration 2.

Schedule				
thought_recording	relaxation	cognitive_rehearsal	sleep_hygine	sleep_restriction

Table 6.16: *Schedule* step 4 iteration 2.

- Step 5, length(s_d) $= 0$ and length(s_i) $= 0$ which satisfies the if condition at line 21. Hence, the Scheduler sends the message "All exercises are scheduled" to the coaches.

Algorithm 2 Scheduling Conflicting Exercises

```
1: procedure SCHEDULER($s_d, s_i$)
2:     l ← length(Schedule)
3:     while $length(s_i) > 0$ and $length(s_d) > 0$ do
4:         procedure RES_CON_DEC($\alpha_i, \alpha_d$)
5:             if $\alpha_{i_1}.t_s + \alpha_{i_1}.l \leq \alpha_{d_1}.t_s$ then
6:                 return $\alpha_{cand} \leftarrow \alpha_{i_1}$          ▷ $\alpha_{cand}$ is the candidate exercise
7:             else if $\alpha_{d_1}.t_s + \alpha_{d_1}.l \leq \alpha_{i_1}.t_s$ then
8:                 return $\alpha_{cand} \leftarrow \alpha_{d_1}$
9:             else
10:                return $\alpha_{cand} \leftarrow$ Resource_Conflict_Res($\alpha_{i_1}, \alpha_{d_1}$)
11:            end if
12:        end procedure
13:        if length(Schedule) = 0 then
14:            insert_Schedule($\alpha_{cand}$)
15:            remove($\alpha_{cand}, s_{cand}$)          ▷ remove the exercise from the
   local_schedule that has the same planner as $\alpha_{cand}$
16:        else
17:            procedure KNOW_CON_DEC($\alpha_{cand}$)
18:                while length(Schedule) > 0 do
19:                    if $\alpha_{cand}.p = Schedule[l].p$ then
20:                        $l - -$;          ▷ l decrements by 1
21:                    else
22:                        for for each p $\in \alpha_{cand}.pre$ do
23:                            if $\neg p \in Schedule[l].post$ then
24:                                $\alpha_{cand1} < -$Knowledge_Conflict_Res($\alpha_{cand}, Schedule[l]$)
25:                            end if
26:                        end for
27:                    end if
28:                end while
29:            end procedure
30:            insert_Schedule($\alpha_{cand1}$)
31:            remove($\alpha_{cand1}, s_{cand1}$)
32:        end if
33:    end while
34:    if $length(s_i) = 0$ and $length(s_d) > 0$ then
35:        for $k = 1; k <= length(s_d); k + +$ do
36:            insert_Schedule($\alpha_{d_k}$)
37:            remove($s_d, \alpha_{d_k}$)
38:        end for
```

```
39:     else if $length(s_i) > 0$ and $length(s_d) = 0$ then
40:         for $j = 1; j <= length(s_i); j++$ do
41:             insert_Schedule($\alpha_{i_j}$))
42:             remove($s_i, \alpha_{i_j}$)
43:         end for
44:     else
45:         print("All exercises are scheduled")
46:     end if
47: return Schedule
48: end procedure
```

6.3.2 Scheduling conflicting exercises

Conflict detection is the triggering event for conflict resolution. In algorithm 2, we outline the procedure of how the *Scheduler* would generally function if there are conflicts between the coaches while integrating their *local_schedule*s. This algorithm (algorithm 2) is based on two assumptions: i) unlike the previous algorithm where we assumed that all the exercises are consistent with each other, this time we only assume that all the exercises of a *local_schedule* are consistent with each other; ii) in a *local_schedule*, exercises are sorted according to their start times. Like the previous algorithm, this time too the Scheduler schedules the first exercise of a *local_schedule* that holds the lowest start time in that list. Below the algorithm 2 is described in words.

- Line 1: the Scheduler procedure takes s_i and s_d as inputs.
- Line 2: Assigns the length of the *Schedule* in a variable, l
- Line 3: Checks the lengths of s_i and s_d. If length(s_i) and length(s_d) are greater than 0 which means there are exercises in the local_schedules to be scheduled on the Schedule, the process will go into the conflict detection procedure, i.e., Res_Con_Dec at line 4.
- Line 4: α_{d_1} and α_{i_1} of s_i and s_d which have the lowest start time in the lists are taken as input into the Res_Con_Dec procedure.
- Lines 5,6: Suppose, α_{i_1} has the lower start time between the two exercises, and it also ends before its next exercise starts, i.e., its start time plus duration is lower or equal to the start time of the next exercise that means the exercises do not conflict and α_{i_1} is returned to the main procedure as the candidate exercise to be scheduled in this iteration.
- Lines 7,8: Same as lines 5,6.

- Lines 9,10: If any of the conditions at line 5 or 7 are not satisfied that means the exercises are in conflict. And they are passed to the conflict resolution procedure, i.e., Resource_Conflict_Res. And the resolution of this procedure are returned as the candidate exercise at line 10.
- Lines 13 - 15: If there is no exercise on the Schedule, i.e., length(Schedule) = 0, the candidate exercise will be scheduled at line 14 and removed from its local_schedule at line 15.
- Line 16: If length(Schedule) $>$ 0, the process enters into the knowledge conflict detection process, i.e., Know_Con_Dec procedure at line 17 which takes the candidate exercise as input.
- Lines 18 to 28: The conflict search occurs between the candidate exercise and the scheduled exercises from the most recent scheduled exercise to the earliest one on the Schedule. Schedule[l] gives the l^th exercise or the most recent exercise on the Schedule, i.e., Schedule[5] gives the fifth exercise on the Schedule from its start point. Unlike length, index starts from 1. There is nothing like Schedule[0]. Such expression will throw exceptions like "invalid". At line 18, the while loop executes as long as the value of l is greater than 0 so that the candidate exercise can be compared with all the exercises on the Schedule. At line 19, the Scheduler checks the planners of the exercises. If Schedule[l] has the same planner as the candidate exercise, the Scheduler will move to the Schedule[l-1] exercise. Otherwise, it will move to the for loop at line 22 where for each atomic proposition of the precondition of the candidate exercise, the complement of the proposition is searched in the postconditions of the former exercises. If knowledge conflict is detected, the two exercises are passed to the Knowledge_Conflict_Res procedure to resolve the conflict.

6.4 Resource Conflict Resolution

If resource conflict is detected between two exercises, it can be resolved by modifying one of the conflicting exercise's start time, i.e., preponing or postponing the exercise. But the question is whose start time should be modified? It is very important to identify the exercise whose modification will not affect the outcome of the coaching process. We introduce two new attributes to the definition of an exercise named *movabilityRange* and *shortener* which will help the *Scheduler* to prioritize the exercises in case of resource conflicts.

- **movabilityRange**: It is a variable that specifies a time period by which an exercise can be preponed or postponed on the Schedule. For instance, InsomniaCoach plans the exercise of filling in the sleep diary at 10:00, with duration 20 minutes, and *movabilityRange* 60 minutes. On the other hand, DepressionCoach plans the exercise of cognitive rehearsal at 10:15 with a duration of 15 minutes and *movabilityRange* of 5 minutes. The exercises conflict each other with respect to time. In this case, the *movabilityRange* of the exercises can guide the Scheduler to resolve the conflict between the coaches. The insomnia exercise has the *movabilityRange* of 60 minutes which is much higher than the *movabilityRange* of the depression exercise. The Scheduler can move the insomnia exercise forward by 5 minutes and free the coaches from the conflict. If the *movabilityRanges* of the two exercises are equal or an exercise's *movabilityRange* is not enough to resolve the conflict, then, both of the coaches can contribute their best to resolve the conflict by moving their exercises forward or backward by the *movabilityRange*s of both of the exercises.

- **shortener**: The coachee may not need the whole allotted time to perform an exercise. Or sometimes she may need more than the planned duration to accomplish an exercise, i.e., the coachee may finish filling in the sleep diary earlier but need more time to record her thoughts. Such issues may create conflicts between the e-coach and the coachee in the execution time of the exercises or when the coachee will perform the exercises. To avoid such conflicts, the coaches may set an overestimated value to the duration like 5 or 10 minutes more than the actual duration which would eventually prevent some conflicts that could occur the execution time. For instance, for the exercise of recording thoughts, the DepressionCoach sets $l = 25$ where in reality it requires only 20 minutes to finish. However, it gives the coachee some time to relax. Besides, it may also boost her confidence and self-belief that she can accomplish a job before the planned end time. It also helps the coaches to resolve the conflict when *movabilityRange* of two exercises are not sufficient to resolve the conflict. In this case, shortening the duration of one of the exercises to its original duration may resolve the conflict.

- Ask the coaches: If the above two resolution techniques do not resolve the conflict, the Scheduler will send the exercises to their coaches to modify their start time, duration, movabilityRange or shortener.

- Line 1: The procedure Resource_Conflict_Res takes the two exercises that are in conflict.

- Lines 2,3: the exercise that has the lower start time between the two is assigned to the variable α_{for} by using the method minimum. And the other exercise is assigned to the variable α_{lat}.

Algorithm 3 Resolution of resource conflicts

```
 1: procedure RESOURCE_CONFLICT_RES($\alpha_i, \alpha_d$)
 2:     $\alpha_{for} \leftarrow \min(\alpha_i.t_s, \alpha_d.t_s)$
 3:     $\alpha_{lat} \leftarrow \max(\alpha_i.t_s, \alpha_d.t_s)$
 4:     overlap $\leftarrow (\alpha_{for}.t_s + \alpha_{for}.l) - \alpha_{lat}.t_s$
 5:     if $\alpha_{lat}.mov \geq overlap$ then        ▷ mov is short form of movabilityRange
 6:         $\alpha_{lat}.t_s \leftarrow \alpha_{lat}.t_s + overlap$
 7:     else if $\alpha_{for}.mov \geq$ overlap then
 8:         if $\alpha_{for}.t_s$ - overlap $\leq Schedule[l].(t_s + l)$ then
 9:             $\alpha_{for}.t_s \leftarrow \alpha_{for}.t_s - overlap$
10:         end if
11:     else if $\alpha_{for}.mov + \alpha_{lat}.mov \geq$ overlap then
12:         if $\alpha_{for}.(t_s - mov) \leq Schedule[l].(t_s + l)$ then
13:             $\alpha_{lat}.t_s \leftarrow \alpha_{lat}.t_s + mov$
14:             $\alpha_{for}.t_s \leftarrow \alpha_{for}.(t_s - mov)$
15:         else
16:             if $\alpha_{for}.short + \alpha_{lat}.mov \geq$ overlap then
17:                 $\alpha_{for}.l \leftarrow \alpha_{for}.(l - short)$
18:                 $\alpha_{lat} \leftarrow \alpha_{lat}.(t_s + mov)$
19:             end if
20:         end if
21:     else if $\alpha_{lat}.short \geq$ overlap then        ▷ short is short form of shortener
22:         $\alpha_{lat}.l \leftarrow \alpha_{lat}.l - short$
23:         return $\alpha_i$
24:     else if $\alpha_{for}.short \geq$ overlap then
25:         $\alpha_{for}.l \leftarrow \alpha_{for}.l - short$
26:         return $\alpha_d$
27:     else if $\alpha_{for}.short + \alpha_{lat}.short \geq$ overlap then
28:         $\alpha_{lat}.l \leftarrow \alpha_{lat}.l - short$
29:         $\alpha_{for}.l \leftarrow \alpha_{for}.l - short$
30:     else
31:         send_message("modify the start times of the exercises.")
32:     end if
33:     return $\alpha_{for}$
34: end procedure
```

- Line 4: overlap is another local variable that is the value of the time by which the exercises overlap each other.
- Line 5,6: If α_{lat}'s *movabilityRange* is bigger than the overlap, then the conflict can be resolved by moving its start time to $\alpha_{lat}.(t_s + overlap)$.
- Lines 7 to 10: If α_{for} has the movabilityRange greater than the overlap, then

we can shift it too by doing $\alpha_{for}.t_s \leftarrow \alpha_{for}.t_s -$overlap. However, changing its start time may create conflict between it and the latest exercise on the Schedule which would be the nearest exercise if α_{for} gets scheduled. So the Scheduler looks for conflict at line 8. If $\alpha_{for}.t_s-$ overlap is less than or equal to the end time of the exercise at Schedule[l], then its start time will be changed to $\alpha_{for}.t_s-$ overlap to resolve the conflict.

- Lines 11 to 20: If an exercise's movabilityRange is not enough to resolve the conflict, the Scheduler checks the sum of the exercises' mov value. If the sum is greater than or equal to the overlap, then both exercises are moved to reolve the conflict. However, to move α_{for}, it must check that the move will not cause conflict between α_{for} and the exercise at Schedule[l]. If α_{for} will conflict with Schedule[l] after the move, the Scheduler checks the shortener of α_{for}. If shortening the duration of α_{for} and moving α_{lat} resolve the conflict, then α_{for} will be shortened and α_{lat} will be moved at line 17 and 18.

- Line 21 to 26: If the above conditions do not satisfy, the shorteners' values are checked. Shortening the exercise of an exercise is less preferable than changing the start time of an exercise because shortening an exercise or giving exact time to the coachee to finish an exercise may pressurize the coachee. Hence, the Scheduler opts for it at the end. If an exercise's shortener's value is greater than the overlap, the exercise is shortened to resolve the exercise at line 22 or 25. However, if an exercise's shortener's value is not enough to resolve the conflict, the Scheduler will check the sum of their shorteners' value. If it is greater than the overlap, both exercises are shortened as much as possible to resolve the conflict.

- Line 31: If the conflict cannot be resolved, the Scheduler sends message to the coaches to modify the start times of α_{lat} as modifying one of the exercise's start time resolve the conflict. As we already said the Scheduler always schedules the exercise with the lower start time. Hence, it asks the coach of α_{lat} to modify its start time.

- Line 31: After resolving the conflict, it returns α_{for} as the candidate exercise.

6.5 Knowledge Conflict Resolution

Knowledge conflict resolution is perhaps the most complicated aspect of this thesis. We tried to resolve such conflicts in a simple and effective way by changing start times of the exercises. Below the resolution schemes for knowledge conflicts are described.

I **Swapping start times:** If an exercise's precondition can be invalidated due to the postcondition of another exercise, then that exercise's postcondition could also be invalidated by some other exercise's postcondition. For instance, dancing as a distraction exercise of CBT-D has the postcondition {excitement}. While its following exercise sleeping from the CBT-I has the precondition {calmness}. Thus the two exercises are in conflict. However, there is relaxation exercise in CBT-I which has the postcondition {calmness} and it invalidates the postcondition of the distraction exercise. So if the start times of the two exercises, i.e., sleeping and relaxation exercise are swapped, the knowledge conflict between dancing and sleeping will be resolved.

II **Supportive Activity**: This technique is designed to resolve the knowledge conflict during the execution phase, i.e., when the coachee starts performing the exercises. In this case, the coaches or the e-coaching system does not resolve the conflict on its own, but takes opinion from the coachee. The coaches and the coachee can resolve their conflict by adding supportive activities in between the conflicting exercises that could change the affect and satisfy the requirement. For instance, the conflict between the physical exercise planned by DepressionCoach and relaxation exercise planned by InsomniaCoach can be resolved by postponing relaxation exercise and adding a "go home" activity before the relaxation exercise. However, scheduling a supportive activity at the planning phase without talking to the coachee may not be the best resolution technique as the coachee may not be always available to do some additional activities along with the therapeutic exercises. For instance, the coachee may not be able to go home right after doing physical exercise because of body ache or rain. Hence taking opinions from the coachee and thus adding supportive activity would be useful, for instance, the coachee could suggest the coaches that instead of going home she would do the relaxation exercise in a quiet place of the gym. This way conflict is resolved.

The resolution of knowledge conflicts is outlined in algorithm 4. Below we describe the algorithm in a step by step manner.

- Line 1: The Knowledge_Conflict_Res is the procedure for resolving knowledge conflicts that occur between a candidate exercise and an already scheduled exercise. Let's assume between the two parameters α_x is the candidate exercise and *Schedule[v]* is the already scheduled exercise.
- Lines 2 to 6: The candidate exercise's source, i.e., its local schedule is identified. And a variable named s_x is used to hold all the exercises from s_x except α_x at line 3 and 5.

Algorithm 4 Resolution of knowledge conflicts

```
procedure KNOWLEDGE_CONFLICT_RES($\alpha_x$, $Schedule[v]$)
    if $\alpha_x.p == i$ then
        $s_x \leftarrow s_i - \alpha_x$
    else
        $s_x \leftarrow s_d - \alpha_x$
    end if
    for each $\alpha$ in $s_x$ do
        for each $p \in \alpha_x.pre$ do
            if $p \in \alpha_{post}$ then
                $\alpha_{temp} \leftarrow \alpha$
                for i=length(Schedule); i>0;i– do
                    for each $q \in \alpha_{temp}.pre$ do
                        if $\neg q \notin Schedule[i].post$ then
                            switch_start_time($\alpha_{x_1}$, $\alpha_{temp}$)
                        end if
                    end for
                end for
            end if
        end for
    end for
    return $\alpha_{x_1}$
end procedure
```

- Line 7 If there is an exercise in s_x that satisfies the precondition of α_x by its postcondition, the Scheduler assigns that exercise into a temporary variable α_{temp} to check whether α_{temp} has any conflicts with the scheduled exercises. If it does not, then the start times of α_x and α_{temp} are switched.
- The procedure returns the new α_x which does not have any resource or knowledge conflict with other exercises.

However, if there is no such exercise like α_{temp}, the Scheduler asks the coaches to negotiate with the coachee in the execution phase.

Example 6.5.1. We take two sample local_schedules, i.e., s_d and s_i to describe the algorithm 2 which also includes algorithm 3 and 4 as sub-processes at lines 10 and 24. We assume that the exercises are consistent internally. Therefore, the exercises of the same domain do not conflict with each other. Also, the exercises in a local schedule are sorted by their start times. Each exercise has seven attributes which include 1) start time or t_s, 2) duration or l, 3) preconditions or pre, 4) postconditions or $post$, 5) movabilityRange or mov, 6) shortener or $short$, and 7) planner or p. Besides, there is no exercise scheduled on the Schedule at this moment.

s_d							
α_d	t_s	l	pre	$post$	mov	$short$	p
α_{d_1}	08:00	15	{a}	{b}	4	2	d
α_{d_2}	12:25	20	{c}	{d}	7	5	d

Table 6.17: Local Schedule of DepressionCoach (s_d)

s_i							
α_i	t_s	l	pre	$post$	mov	$short$	p
α_{i_1}	08:10	25	{¬ b}	{d}	1	1	i
α_{i_2}	11:45	20	{c}	{¬b}	7	5	i

Table 6.18: Local Schedule of InsomniaCoach (s_i)

- Line 1: the Scheduler takes s_d and s_i as the input parameters where $s_d = \{\alpha_{d_1}, \alpha_{d_2}\}$ and $s_i = \{\alpha_{i_1}, \alpha_{i_2}\}$
- Line 2: it assigns the length of the Schedule in the variable l. Currently, there is no exercise scheduled on the Schedule. So, length(Schedule) is 0 which means $l = 0$ also.
- Line 3: the while loop starts and it continues until the lengths of both s_d and s_i are greater than 0. Currently, length(s_d)=3, length(s_i)=2, so the while loop will iterate for 2 times.
  - Iteration 1, line 4: the process *Res_Con_Dec* starts by taking the first two exercises of s_d and s_i as input parameters ,i.e., α_{d_1} and α_{i_1}. This process will return the candidate exercise to be scheduled.
  - Iteration 1, line 5: here, $\alpha_{i_1}.t_s$ = 8:10, $\alpha_{i_1}.l$ = 8:10, $\alpha_{d_1}.t_s$ = 8:00. The if condition is false as $\alpha_{i_1}.t_s + \alpha_{i_1}.l \nleq \alpha_{d_1}.t_s$.
  - Iteration 1, line 7: the else if condition is also false as $\alpha_{d_1}.t_s + \alpha_{d_1}.l \nleq \alpha_{i_1}.t_s$.
  - Iteration 1, line 10: since the conditions at lines 5 and 7 are not true that means α_{d_1} and α_{i_1} are in conflict. In that case, the exercises will be passed to the *Resource_Conflict_Res* to resolve the resource conflict between the exercises. And *Resource_Conflict_Res* returns the α_{cand} after resolving the conflict.
    - Lines 2,3: the process *Resource_Conflict_Res* determines between α_{d_1} and α_{i_1} which exercise will be scheduled first and which one latter by judging their start times. The exercise with the lower start time is assigned to a local variable named α_{for} and the other exercise is assigned to another local variable named α_{lat}. Here, $\alpha_{for} = \alpha_{d_1}$ and $\alpha_{lat} = \alpha_{i_1}$.

    * Line 4: the time overlap is measured at this line. Here, overlap = 5 minutes.
    * Line 5: $alpha_{lat}.mov$ = 1 which is lower than the overlap. So the if condition at this line is false.
    * Line 7: $alpha_{for}.mov$ = 4. So the else if condition at this line is also false.
    * Line 11: $alpha_{for}.mov + alpha_{lat}.mov$ =5 which is equal to the overlap. So the else if condition of this line is true.
        · Line 13: α_{for} is preponed to $\alpha_{for}.t_s - mov$ which is 7:56.
        · Line 14: α_{lat} is postponed by changing its start time to $\alpha_{lat}.t_s + \alpha_{lat}.mov$ which is 8:11.
    * Line 33: the process returns α_{for} to be the candidate exercise
  – Iteration 1, line 10: $\alpha_{cand} = \alpha_{d_1}$.
  – Iteration 1, line 13: length(Schedule) is 0.
  – Iteration 1, line 14: the process insert_Schedule schedules α_{cand} on the Schedule as the condition at line 13 is true.
  – Iteration 1, line 15: α_{cand} is removed from the local_schedule it belongs to. Here α_{cand} is actually α_{d_1}. So α_{d_1} is removed from s_d. And length(s_d) reduces to 1.

s_d							
α_d	t_s	l	pre	$post$	mov	$short$	p
~~α_{d_1}~~	~~07:56~~	~~15~~	~~{a}~~	~~{b}~~	~~0~~	~~2~~	~~d~~
α_{d_2}	12:25	20	{c}	{d}	7	5	d

Table 6.19: Local Schedule of DepressionCoach (s_d)

s_i							
α_i	t_s	l	pre	$post$	mov	$short$	p
α_{i_1}	08:11	25	{¬ b}	{d}	0	1	i
α_{i_2}	11:45	20	{c}	{¬b}	7	5	i

Table 6.20: Local Schedule of InsomniaCoach (s_d)

▪ Line 3: length(s_d) is 1 and length(s_i) is 2. The while loop continues to the next iteration.

  – Iteration 2, line 4: The process *Res_Con_Dec* starts by taking the first two exercises of s_d and s_i as input parameters ,i.e., α_{d_2} and α_{i_1}.
  – Iteration 2, line 5: Here, $\alpha_{i_1}.t_s$ = 8:11, $\alpha_{i_1}.l$ = 25, $\alpha_{d_2}.t_s$ = 12:25. The if condition is true as $\alpha_{i_1}.t_s + \alpha_{i_1}.l \leq \alpha_{d_2}.t_s$.

- Iteration 2, line 6: α_{i_1} becomes the candidate exercise, i.e., $\alpha_{cand} = \alpha_{i_1}$
- Iteration 2, line 13: length(Schedule) is 1 so this if condition is false.
- Iteration 2, line 16,17: The process enters into the sub-process *KNOW_CON_DEC* which takes α_{cand} as the input parameter.
- Iteration2, line 18: length(Schedule) is 1 which is greater than 0. So the while loop starts.
    * Iteration2, Line 19: l=length(Schedule) =1 and Schedule[1]= α_{d_1}. The planner of α_{cand} is p and the planner of α_{d_1} is d. Hence, the if condition at this line is false.
    * Iteration2, Lines 22-: The precondition of α_{cand} includes $\{\neg b\}$ and the postcondition of Schedule[1] or α_{d_1} is $\{b\}$. Since the condition at line 23 is true, the sub-process Knowledge_Conflict_Res is invoked at line 24. The process Knowledge_Conflict_Res takes α_{cand} and Schedule[1] which are α_{i_1} and α_{d_1}.
        - Line 1: α_x is α_{i_1} and α_{d_1} is Schedule[v].
        - Line 2: Since the planner of α_x is i, s_i minus α_i is assigned to the local variable s_x.
        - Lines 7-20: The precondition of α_x is $\{\neg b\}$ which is searched in the postconditions of the exercises from s_x. There is only one exercise in s_x that is α_{i_2} which has $\{\neg b\}$ as its postcondition. α_{i_2} is assigned to a local variable α_{temp}. Then the negation of the precondition of α_{temp} which is $\{\neg c\}$ is searched in the postcondition of the already scheduled exercise α_{d_1}. If it is not found, the start times of α_x and α_{temp} are swapped and the new α_x which is α_{i_2} is returned to the main process.
    * Iteration2, Line 24: the returned value of the process Knowledge_Conflict_Res is assigned to the variable α_{cand1}.
    * Iteration2, Line 30: α_{cand1} which is α_{i_2} is scheduled on the Schedule and removed from s_i.

s_d							
α_d	t_s	l	pre	$post$	mov	$short$	p
~~α_{d_1}~~	~~07:56~~	~~15~~	~~{a}~~	~~{b}~~	~~4~~	~~2~~	~~d~~
α_{d_2}	12:25	20	{c}	{d}	7	5	d

Table 6.21: Local Schedule of DepressionCoach (s_d)

- In the next two iterations α_{i_2} and α_{d_2} will be scheduled after following the same processes as described above. These two exercises do not have resource conflict or knowledge conflict with each other. Therefore their scheduling

s_i							
α_i	t_s	l	pre	$post$	mov	$short$	p
~~α_{i_2}~~	~~08:11~~	~~20~~	~~{c}~~	~~{¬b}~~	~~7~~	~~5~~	~~i~~
α_{i_1}	11:45	25	{¬ b}	{d}	1	1	i

Table 6.22: Local Schedule of InsomniaCoach (s_d)

process is rather straightforward. After scheduling the four exercise from s_d and s_i, the Schedule looks like table 6.23.

Schedule			
$t_s = 7:56$	$t_s = 8:11$	$t_s = 11:45$	$t_s = 12:25$
α_{d_1}	α_{i_2}	α_{i_1}	α_{d_2}

Table 6.23: *Schedule* step 4 iteration 2.

6.6 Conclusion

In this chapter, we have discussed the classification of conflicts that may occur between the coaches in the MHC, i.e., resource conflicts and knowledge conflicts. We have also modeled the conflict detection and resolution mechanisms to handle the conflicts. We have also showed how the conflict resolution model works via a test case. In the next chapter, we will conclude the thesis.

Chapter 7

Conclusion

The principal problem that is addressed in this thesis was *how to resolve conflicts between agents in a MAS*. We have approached the problem within the domain of an e-coaching system, the so-called MHC that offers e-coaching for multiple disorders, i.e., depression and insomnia. MHC is built upon the MAS paradigm and it consists of two expert coaches, i.e., DepressionCoach and InsomniaCoach who have expertise in depression and insomnia. Besides, there are two other agents, i.e., IOManager and Scheduler who support the coaches to make interactions and plan the coaching process for the coachee. The system offers three kinds of coaching based on the symptoms of the coachee, i.e., depression coaching, insomnia coaching, and depression + insomnia coaching.

In order to deliver coaching on both depression and insomnia, the DepressionCoach and InsomniaCoach cooperate and jointly plan the coaching process. In this process, the coaches individually plan the exercises and then share them with each other via the Scheduler agent. The Scheduler integrates the plans and if there is no conflict between the exercises of different domains he enters the integrated plans on the Schedule tool to share the coaching plans with the coachee. Due to the differences in domain knowledge and scarcity of resources the coaches may get involved in either knowledge conflicts or resource conflicts. The knowledge conflict refers to a situation where a preceding exercise negative affects its following exercise. In other words, if an exercise's precondition gets invalidated by the postconditions of one of its preceding exercises on the Schedule, we refer to this situation as a knowledge conflict. On the other hand, the resource conflict occurs when there is a time overlap between any two exercises of different domains.

The Scheduler detects the conflicts by comparing the start times, duration, preconditions, and postconditions of the exercises before scheduling them. We have proposed a simple technique to resolve the conflicts, and the technique is rescheduling one or both of the exercises that are in conflict. We have shown in examples that if two exercises' times overlap, then changing the start time or duration of one of the exercises is the most basic yet effective way to resolve the resource conflict. For this purpose, we have included two additional attributes in the definition of an exercise which are movabilityRange and shortener. These two attributes guide the Scheduler on how much he can prepone and postpone an exercise on the Schedule. Also, the Scheduler can resolve the knowledge conflict between any two exercises of different domains by rescheduling them.

For our current framework where there are not too many conflicts between the exercises of CBT-I and CBT-D, the model (algorithm 2) would work effectively. However, in the future, if we incorporate other domains of treatments, we have to be conscious while defining exercises in the knowledge-bases of the coaches as the attributes of each exercise play important roles in the resolution process. Nevertheless, the current model works for scheduling exercise for a day. However, it can extended to scheduling exercises for n-weeks by changing the specification of the time-attribute. Currently the time attribute is in the form of hour: minute: second. So the Scheduler schedules exercises for only 24 hours. To extend it to n-weeks or n-months, we need to modify the time -attribute accordingly.

7.1 Strong and weak points of the Solution

The strong points of our conflict resolution model include:

- The Scheduler uses a simple yet useful technique to resolve conflicts in the scope of the scheduling technique;
- The detection and resolution processes of resource and knowledge conflicts work fast as algorithms 3 and 4 take polynomial times. As a result, the entire scheduling process of the Scheduler, i.e., algorithm 2 also takes polynomial time.

The weak points of the resolution mechanism include:

- The success of the technique depends on the attributes of the exercises. If the attributes, i.e., movabilityRange, shortener, preconditions, postconditions are not defined in the desired way, it may not provide the expected result.
- Furthermore, the propositions of the preconditions and postconditions should be represented in a way that the Scheduler can understand. Currently, the Scheduler only understands the preconditions or postconditions of an exercise as the set of comma seperated atomic propositions, i.e., $\{a, b, \neg c\}$. This way he can only handle the negation, and logical AND relation between the atomic propositions. However, if the atomic propositions share any other binary relations such as logical OR, XOR the Scheduler may throw errors.

However, in any case, we can use this model as an initial step towards the development of an e-coaching system that can support multiple disorders.

7.2 Future Work

In the future, we need to implement the model since it is modeled only in papers so there can be still some limitations and unknown facts. After implementing the model, we need to operate a trial to see how the e-coaching system works for this two domains. If we get success with the two domains, we can extend the framework by adding other related domains, i.e., anxiety disorder, substance use, etc.

7.3 What have I learned?

The thesis has been a meaningful experience in my life which not only extended my knowledge in different fields, i.e., life coaching, e-coaching, cognitive behavioral therapy, computer science and multi-agent system but it also developed me as an individual. It has also taught me how computer science, artificial intelligence, and psychology and/or medical science can be combined to develop a healthcare application to help the people live a healthful and joyful life. It made me think about why we should look at the simple techniques to resolve any problems rather than complex ones. I have also learned how to write a thesis effectively and coherently that can deliver its message to the readers.

Bibliography

[1] Research 2 Guidance, "mhealth app developer economics 2016," 2016. Accessed: 2016-10-26. (Archived by WebCite® at http://www. webcitation.org/6lY0vJ78i).

[2] S. M. Haffner, S. Lehto, T. Rönnemaa, K. Pyörälä, and M. Laakso, "Mortality from coronary heart disease in subjects with type 2 diabetes and in nondiabetic subjects with and without prior myocardial infarction," *New England journal of medicine*, vol. 339, no. 4, pp. 229–234, 1998.

[3] N. Tsuno, A. Besset, and K. Ritchie, "Sleep and depression.," *The Journal of clinical psychiatry*, 2005.

[4] K. A. Kaplan and A. G. Harvey, "Hypersomnia across mood disorders: a review and synthesis," *Sleep Medicine Reviews*, vol. 13, no. 4, pp. 275–285, 2009.

[5] P. H. Kleinman, E. D. Wish, S. Deren, and G. A. Rainone, "Multiple drug use: A symptomatic behavior," *Journal of psychoactive drugs*, vol. 18, no. 2, pp. 77–86, 1986.

[6] W. Alshabi, S. Ramaswamy, M. Itmi, and H. Abdulrab, "Coordination, cooperation and conflict resolution in multi-agent systems," in *Innovations and advanced techniques in computer and information sciences and engineering*, pp. 495–500, Springer, 2007.

[7] A. Y. Tang and G. S. Basheer, "A conflict resolution strategy selection method (confrssm) in multi-agent systems," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 5, pp. 398–404, 2017.

[8] T. Wagner, J. Shapiro, P. Xuan, and V. Lesser, "Multi-level conflict in multi-agent systems," in *Proc. of AAAI Workshop on Negotiation in Multi-Agent Systems*, vol. 15, 1999.

[9] R. J. Beun, F. Griffioen-both, R. Ahn, S. Fitrianie, and J. Lancee, "Modeling interaction in automated e-coaching a case from insomnia therapy," 2014.

[10] R. Ahn, “Basic sleep coach knowledge architecture,” 11 2013.

[11] Diabetes.co.uk: the global diabetes community, “Diabetes and depression,” 2018. Accessed: 2018-06-28.

[12] R. J. Beun, S. Fitrianie, F. Griffioen-Both, S. Spruit, C. Horsch, J. Lancee, and W.-P. Brinkman, “Talk and tools: the best of both worlds in mobile user interfaces for e-coaching,” *Personal and Ubiquitous Computing*, vol. 21, no. 4, pp. 661–674, 2017.

[13] S. Cammarata, D. McArthur, and R. Steeb, “Strategies of cooperation in distributed problem solving,” in *Readings in Distributed Artificial Intelligence*, pp. 102–105, Elsevier, 1988.

[14] Y. Demazeau and J.-P. Müller, *Decentralized AI, 2- Proceedings of te Second European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds(MAAMAW-90)*. Elsevier, 1991.

[15] E. H. Durfee, V. R. Lesser, and D. D. Corkill, “Cooperative distributed problem solving,” *The handbook of artificial intelligence*, vol. 4, no. 1, pp. 83–147, 1989.

[16] J. Ferber, *Multi-agent systems: an introduction to distributed artificial intelligence*, vol. 1. Addison-Wesley Reading, 1999.

[17] E. H. Durfee and V. R. Lesser, “Negotiating task decomposition and allocation using partial global planning,” in *Distributed artificial intelligence*, pp. 229–243, Elsevier, 1998.

[18] J. S. Santos, J. O. Zahn, E. A. Silvestre, V. T. Silva, and W. W. Vasconcelos, “Detection and resolution of normative conflicts in multi-agent systems: a literature survey,” *Autonomous agents and multi-agent systems*, vol. 31, no. 6, pp. 1236–1282, 2017.

[19] L. Green, L. Oades, and A. Grant, “Cognitive-behavioral, solution-focused life coaching: Enhancing goal striving, well-being, and hope,” *The Journal of Positive Psychology*, vol. 1, no. 3, pp. 142–149, 2006.

[20] Australian Psychological Society, “Coaching psychology interest group mission statement.” `http//www.psychology.org.au/units/`, 2003. [online; accessed in November,2003].

[21] C. A. Douglas and C. D. McCauley, “Formal developmental relationships: A survey of organizational practices,” *Human Resource Development Quarterly*, vol. 10, no. 3, pp. 203–220, 1999.

[22] D. C. Feldman, “Career coaching: What hr professionals and managers need to know.,” *Human Resource Planning*, vol. 24, no. 2, 2001.

[23] A. M. Grant, “The impact of life coaching on goal attainment, metacognition and mental health,” *Social Behavior and Personality: an international journal*, vol. 31, no. 3, pp. 253–263, 2003.

[24] R. Biswas-Diener, “Personal coaching as a positive intervention,” *Journal of clinical psychology*, vol. 65, no. 5, pp. 544–553, 2009.

[25] P. Williams and D. C. Davis, *Therapist as life coach: Transforming your practice.* WW Norton & Company, 2002.

[26] E. Klinger, *Meaning and void: Inner experience and the incentives in peoples lives.* U of Minnesota Press, 1977.

[27] K. M. Sheldon, T. Kasser, K. Smith, and T. Share, “Personal goals and psychological growth: Testing an intervention to enhance goal attainment and personality integration,” *Journal of Personality*, vol. 70, no. 1, pp. 5–31, 2002.

[28] R. M. Ryan and E. L. Deci, “On happiness and human potentials: A review of research on hedonic and eudaimonic well-being,” *Annual review of psychology*, vol. 52, no. 1, pp. 141–166, 2001.

[29] F. F. Chen, Y. Jing, A. Hayes, and J. M. Lee, “Two concepts or two approaches? a bifactor analysis of psychological and subjective well-being,” *Journal of Happiness Studies*, vol. 14, no. 3, pp. 1033–1068, 2013.

[30] C. R. Snyder, S. T. Michael, and J. S. Cheavens, “Hope as a psychotherapeutic foundation of common factors, placebos, and expectancies.,” 1999.

[31] C. R. Snyder, *Handbook of hope: Theory, measures, and applications.* Academic press, 2000.

[32] S. D. Hollon and A. T. Beck, “Cognitive and cognitive-behavioral therapies.,” 1994.

[33] K. S. Dobson, *Handbook of cognitive-behavioral therapies.* Guilford Press, 2009.

[34] A. T. Beck, “The past and future of cognitive therapy,” *The Journal of psychotherapy practice and research*, vol. 6, no. 4, p. 276, 1997.

[35] P. M. Salkovskis, *Frontiers of cognitive therapy.* Guilford Press, 1997.

[36] A. C. Butler, J. E. Chapman, E. M. Forman, and A. T. Beck, "The empirical status of cognitive-behavioral therapy: a review of meta-analyses," *Clinical psychology review*, vol. 26, no. 1, pp. 17–31, 2006.

[37] A. T. Beck, "Cognitive therapy: Nature and relation to behavior therapy," *Behavior therapy*, vol. 1, no. 2, pp. 184–200, 1970.

[38] A. Ellis, "Reason and emotion in psychotherapy.," 1962.

[39] D. Meichenbaum, "Cognitive behaviour modification," *Cognitive Behaviour Therapy*, vol. 6, no. 4, pp. 185–192, 1977.

[40] S. G. Hofmann, *An introduction to modern CBT: Psychological solutions to mental health problems.* John Wiley & Sons, 2011.

[41] S. G. Hofmann, A. Asnaani, I. J. Vonk, A. T. Sawyer, and A. Fang, "The efficacy of cognitive behavioral therapy: A review of meta-analyses," *Cognitive therapy and research*, vol. 36, no. 5, pp. 427–440, 2012.

[42] C. M. Morin, R. R. Bootzin, D. J. Buysse, J. D. Edinger, C. A. Espie, and K. L. Lichstein, "Psychological and behavioral treatment of insomnia: update of the recent evidence (1998–2004)," *Sleep*, vol. 29, no. 11, pp. 1398–1414, 2006.

[43] M. M. Ohayon, "Epidemiology of insomnia: what we know and what we still need to learn," *Sleep medicine reviews*, vol. 6, no. 2, pp. 97–111, 2002.

[44] N. Breslau, T. Roth, L. Rosenthal, and P. Andreski, "Sleep disturbance and psychiatric disorders: a longitudinal epidemiological study of young adults," *Biological psychiatry*, vol. 39, no. 6, pp. 411–418, 1996.

[45] D. E. Ford and D. B. Kamerow, "Epidemiologic study of sleep disturbances and psychiatric disorders: an opportunity for prevention?," *Jama*, vol. 262, no. 11, pp. 1479–1484, 1989.

[46] G. E. Simon and M. VonKorff, "Prevalence, burden, and treatment of insomnia in primary care," *American Journal of Psychiatry*, vol. 154, no. 10, pp. 1417–1423, 1997.

[47] J. K. Walsh, "Clinical and socioeconomic correlates of insomnia.," *The Journal of clinical psychiatry*, 2004.

[48] C. M. Morin and C. A. Espie, "Cognitive therapy," *Insomnia: A Clinical Guide to Assessment and Treatment*, pp. 77–99, 2004.

[49] R. R. Bootzin, D. Epstein, and J. M. Wood, "Stimulus control instructions," in *Case studies in insomnia*, pp. 19–28, Springer, 1991.

[50] C. A. Espie, S. J. Inglis, S. Tessier, and L. Harvey, “The clinical effectiveness of cognitive behaviour therapy for chronic insomnia: implementation and evaluation of a sleep clinic in general medical practice,” *Behaviour research and therapy*, vol. 39, no. 1, pp. 45–60, 2001.

[51] A. J. Spielman, P. Saskin, and M. J. Thorpy, “Treatment of chronic insomnia by restriction of time in bed,” *Sleep*, vol. 10, no. 1, pp. 45–56, 1987.

[52] World Health Organization, “Mental health, suicide data.” `http://www.who.int/mental_health/prevention/suicide/suicideprevent/en/`, 2018. [online; accessed in July 10, 2018].

[53] NIMH, “Major depression among adults.” National Institue of Mental Health. Retrieved September 4,2017 from https://www.nimh.nih.gov/health/statistics/prevalence/major-depression-among-adults.shtml.

[54] W. H. Organization *et al.*, “Mental health: a call for action by world health ministers,” *Geneva: World Health Organization, Department of Mental Health and Substance Dependence*, 2001.

[55] K. Kroenke, R. Spitzer, and J. Williams, “The phq-9: Validity of a brief depression severity measure. gen intern med. 2001; 16: 606-13.”

[56] R. J. DeRubeis, S. D. Hollon, J. D. Amsterdam, R. C. Shelton, P. R. Young, R. M. Salomon, J. P. OâĂŹReardon, M. L. Lovett, M. M. Gladis, L. L. Brown, *et al.*, “Cognitive therapy vs medications in the treatment of moderate to severe depression,” *Archives of general psychiatry*, vol. 62, no. 4, pp. 409–416, 2005.

[57] S. D. Hollon, R. J. DeRubeis, M. D. Evans, M. J. Wiemer, M. J. Garvey, W. M. Grove, and V. B. Tuason, “Cognitive therapy and pharmacotherapy for depression: Singly and in combination,” *Archives of general psychiatry*, vol. 49, no. 10, pp. 774–781, 1992.

[58] G. E. Murphy, A. D. Simons, R. D. Wetzel, and P. J. Lustman, “Cognitive therapy and pharmacotherapy: Singly and together in the treatment of depression,” *Archives of General Psychiatry*, vol. 41, no. 1, pp. 33–41, 1984.

[59] R. B. Jarrett, M. Schaffer, D. McIntire, A. Witt-Browder, D. Kraft, and R. C. Risser, “Treatment of atypical depression with cognitive therapy or phenelzine: a double-blind, placebo-controlled trial,” *Archives of general psychiatry*, vol. 56, no. 5, pp. 431–437, 1999.

[60] M. Kovacs, A. J. Rush, A. T. Beck, and S. D. Hollon, “Depressed outpatients treated with cognitive therapy or pharmacotherapy: A one-year follow-up,” *Archives of General Psychiatry*, vol. 38, no. 1, pp. 33–39, 1981.

[61] I. M. Blackburn, K. Eunson, and S. Bishop, “A two-year naturalistic follow-up of depressed patients treated with cognitive therapy, pharmacotherapy and a combination of both,” *Journal of Affective disorders*, vol. 10, no. 1, pp. 67–75, 1986.

[62] A. D. Simons, G. E. Murphy, J. L. Levine, and R. D. Wetzel, “Cognitive therapy and pharmacotherapy for depression: Sustained improvement over one year,” *Archives of General Psychiatry*, vol. 43, no. 1, pp. 43–48, 1986.

[63] M. D. Evans, S. D. Hollon, R. J. DeRubeis, J. M. Piasecki, W. M. Grove, M. J. Garvey, and V. B. Tuason, “Differential relapse following cognitive therapy and pharmacotherapy for depression,” *Archives of general psychiatry*, vol. 49, no. 10, pp. 802–808, 1992.

[64] V. Gloaguen, J. Cottraux, M. Cucherat, and I.-M. Blackburn, “A meta-analysis of the effects of cognitive therapy in depressed patients,” *Journal of affective disorders*, vol. 49, no. 1, pp. 59–72, 1998.

[65] A. Watson, T. Bickmore, A. Cange, A. Kulshreshtha, and J. Kvedar, “An internet-based virtual coach to promote physical activity adherence in overweight adults: randomized controlled trial,” *Journal of medical Internet research*, vol. 14, no. 1, 2012.

[66] H. McNamara, “The rise of e-coaching,” *Training Journal*, pp. 66–70, 2011.

[67] T. Bickmore, A. Gruber, and R. Picard, “Establishing the computer–patient working alliance in automated health behavior change interventions,” *Patient education and counseling*, vol. 59, no. 1, pp. 21–30, 2005.

[68] T. W. Bickmore, D. Schulman, and C. L. Sidner, “A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology,” *Journal of biomedical informatics*, vol. 44, no. 2, pp. 183–197, 2011.

[69] O. A. Blanson Henkemans, P. J. van der Boog, J. Lindenberg, C. A. van der Mast, M. A. Neerincx, and B. J. Zwetsloot-Schonk, “An online lifestyle diary with a persuasive computer assistant providing feedback on self-management,” *Technology and Health Care*, vol. 17, no. 3, pp. 253–267, 2009.

[70] F. Both, P. Cuijpers, M. Hoogendoorn, M. C. Klein, A. Fred, J. Filipe, and H. Gamboa, “Towards fully automated psychotherapy for adults: Bas-behavioral activation scheduling via web and mobile phone,” 2010.

[71] C. A. Espie, S. D. Kyle, C. Williams, J. C. Ong, N. J. Douglas, P. Hames, and J. S. Brown, “A randomized, placebo-controlled trial of online cognitive behavioral therapy for chronic insomnia disorder delivered via an

automated media-rich web application," *Sleep*, vol. 35, no. 6, pp. 769–781, 2012.

[72] G. Valenza, M. Nardelli, A. Lanata, C. Gentili, G. Bertschy, R. Paradiso, and E. P. Scilingo, "Wearable monitoring for mood recognition in bipolar disorder based on history-dependent long-term heart rate variability analysis," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 5, pp. 1625–1635, 2014.

[73] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The knowledge engineering review*, vol. 10, no. 2, pp. 115–152, 1995.

[74] D. C. Dennett, *The intentional stance.* MIT press, 1989.

[75] A. S. Rao and M. P. Georgeff, "Modeling rational agents within a bdi-architecture.," *KR*, vol. 91, pp. 473–484, 1991.

[76] K. S. Barber, T.-H. Liu, and S. Ramaswamy, "Conflict detection during plan integration for multi-agent systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 31, no. 4, pp. 616–628, 2001.

[77] K. Decker and J. Li, "Coordinated hospital patient scheduling," in *Multi Agent Systems, 1998. Proceedings. International Conference on*, pp. 104–111, IEEE, 1998.

[78] C. C. Marinagi, C. D. Spyropoulos, C. Papatheodorou, and S. Kokkotos, "Continual planning and scheduling for managing patient tests in hospital laboratories," *Artificial Intelligence in Medicine*, vol. 20, no. 2, pp. 139–154, 2000.

[79] O. Baujard, V. Baujard, S. Aurel, C. Boyer, and R. Appel, "Marvin, multi-agent softbot to retrieve multilingual medical information on the web," *Medical Informatics*, vol. 23, no. 3, pp. 187–191, 1998.

[80] A. Moreno and D. Isern, "Accessing distributed health-care services through smart agents," in *Proceedings of the 4th IEEE International Workshop on Enterprise Networking and Computing in the Health Care Industry (HealthCom 2002), Nancy, France*, pp. 34–41, 2002.

[81] S. Barro, J. Presedo, D. Castro, M. Fernandez-Delgado, S. Fraga, M. Lama, and J. Vila, "Intelligent telemonitoring of critical-care patients," *IEEE Engineering in medicine and biology magazine*, vol. 18, no. 4, pp. 80–88, 1999.

[82] G. Lanzola, L. Gatti, S. Falasconi, and M. Stefanelli, "A framework for building cooperative software agents in medical applications," *Artificial intelligence in medicine*, vol. 16, no. 3, pp. 223–249, 1999.

[83] J. E. Larsson and B. Hayes-Roth, “Guardian: An intelligent autonomous agent for medical monitoring and diagnosis,” *IEEE Intelligent Systems*, no. 1, pp. 58–64, 1998.

[84] J. Vázquez-Salceda, “Using agent-mediated institutions for the distribution of human tissues among hospitals,” *Advanced Course on Artificial Intelligence-ACAI-01*, pp. 205–209, 2001.

[85] J. Vázquez-Salceda, J. A. Padget, U. Cortés, A. López-Navidad, and F. Caballero, “Formalizing an electronic institution for the distribution of human tissues,” *Artificial Intelligence in Medicine*, vol. 27, no. 3, pp. 233–258, 2003.

[86] A. Moreno, A. Valls, and J. Bocio, “Management of hospital teams for organ transplants using multi-agent systems,” in *Conference on Artificial Intelligence in Medicine in Europe*, pp. 374–383, Springer, 2001.

[87] M. Beer, W. Huang, and A. Sixsmith, “Using agents to build a practical implementation of the inca (intelligent community alarm) system,” in *Intelligent agents and their applications*, pp. 295–328, Springer, 2002.

[88] U. Cortés, R. Annicchiarico, J. Vázquez-Salceda, C. Urdiales, L. Cañamero, M. López, M. Sànchez-Marrè, and C. Caltagirone, “Assistive technologies for the disabled and for the new generation of senior citizens: the e-tools architecture,” *AI Communications*, vol. 16, no. 3, pp. 193–207, 2003.

[89] L. M. Camarinha-Matos, H. Afsarmanesh, *et al.*, “Virtual communities and elderly support,” *Advances in Automation, Multimedia and Video Systems, and Modern Computer Science*, pp. 279–284, 2001.

[90] A. Farías and T. N. Arvanitis, “Building software agents for training systems: a case study on radiotherapy treatment planning,” *Knowledge-Based Systems*, vol. 10, no. 3, pp. 161–168, 1997.

[91] C. H. Horsch, J. Lancee, F. Griffioen-Both, S. Spruit, S. Fitrianie, M. A. Neerincx, R. J. Beun, and W.-P. Brinkman, “Mobile phone-delivered cognitive behavioral therapy for insomnia: a randomized waitlist controlled trial,” *Journal of medical Internet research*, vol. 19, no. 4, 2017.

[92] J. Van Diggelen, *Achieving Semantic Interoperability in Multi-agent Systems.* Universiteit Utrecht, 2007.

[93] J. Rodrıguez, “On the design and construction of agent-mediated electronic institutions, vol. 14 of iiia monographs,” 2001.

[94] C. E. Shannon, “A mathematical theory of communication,” *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.

[95] T. Finin, R. Fritzson, D. McKay, and R. McEntire, “Kqml as an agent communication language,” in *Proceedings of the third international conference on Information and knowledge management*, pp. 456–463, ACM, 1994.

[96] A. Fipa, “Fipa acl message structure specification,” *Foundation for Intelligent Physical Agents, http://www. fipa. org/specs/fipa00061/SC00061G. html (30.6. 2004)*, 2002.

[97] C. F. Ngolah and B. H. Far, “A tutorial on agent communication and knowledge sharing,” *University of Calgary, SENG609*, vol. 22.

[98] J. E. Doran, S. Franklin, N. R. Jennings, and T. J. Norman, “On cooperation in multi-agent systems,” *The Knowledge Engineering Review*, vol. 12, no. 3, pp. 309–314, 1997.

[99] B. Moulin and B. Chaib-Draa, “An overview of distributed artificial intelligence,” *Foundations of distributed artificial intelligence*, vol. 1, pp. 3–55, 1996.

[100] M. Wooldridge, *An introduction to multiagent systems.* John Wiley & Sons, 2009.

[101] R. D. Stolorow, “Toward a pure psychology of inner conflict.,” 1985.

[102] Estimation lemma, “Karl marx — Wikipedia, the free encyclopedia,” 2018. [Online; accessed 28-January-2018].

[103] I. L. Horowitz, *Communicating Ideas: The Politics of Scholarly Publishing.* Routledge, 2017.

[104] J. M. Burns, *Transforming leadership: A new pursuit of happiness*, vol. 213. Grove Press, 2003.

[105] J. A. Aho, *German realpolitik and American sociology: an inquiry into the sources and political significance of the sociology of conflict.* Bucknell University Press, 1975.

[106] R. D. Duvall, “Understanding conflict and war, vol. 2: The conflict helix. by rummel rj.(beverly hills, calif.: Halsted press, 1976. pp. 400. $17.50.),” *American Political Science Review*, vol. 73, no. 3, pp. 951–952, 1979.

[107] C. Tessier, L. Chaudron, and H.-J. Müller, *Conflicting agents: conflict management in multi-agent systems*, vol. 1. Springer Science & Business Media, 2006.

[108] C. Castelfranchi, "Conflict ontology," in *Computational conflicts*, pp. 21–40, Springer, 2000.

[109] V. T. Da Silva and J. Zahn, "Normative conflicts that depend on the domain," in *International Workshop on Coordination, Organizations, Institutions, and Norms in Agent Systems*, pp. 311–326, Springer, 2013.

[110] M. Kollingbaum and T. Norman, "Strategies for resolving norm conflict in practical reasoning," in *ECAI workshop coordination in emergent agent societies*, vol. 2004, 2004.

[111] M. J. Kollingbaum and T. J. Norman, "Informed deliberation during norm-governed practical reasoning," in *International Conference on Autonomous Agents and Multiagent Systems*, pp. 183–197, Springer, 2005.

[112] M. Kollingbaum, T. Norman, A. Preece, and D. Sleeman, "Norm refinement: Informing the re-negotiation of contracts," in *ECAI 2006 Workshop on Coordination, Organization, Institutions and Norms in Agent Systems, COIN@ ECAI*, vol. 2006, pp. 46–51, 2006.

[113] M. J. Kollingbaum, T. J. Norman, A. Preece, and D. Sleeman, "Norm conflicts and inconsistencies in virtual organisations," in *Coordination, organizations, institutions, and norms in agent systems II*, pp. 245–258, Springer, 2007.

[114] K. Barber, T. Liu, and D. Han, "Strategic decision-making for conflict resolution in dynamic organized multi-agent systems," *A Special Issue of CERA Journal*, 2000.

[115] Y. E. Ioannidis and T. K. Sellis, *Conflict resolution of rules assigning values to virtual attributes*, vol. 18. ACM, 1989.

[116] J. S. Rosenschein and G. Zlotkin, *Rules of encounter: designing conventions for automated negotiation among computers.* MIT press, 1994.

[117] H. Emami and K. Narimanifar, "A conflict resolution approach using prioritization strategy," *J Inform Syst Telecom*, vol. 1, pp. 19–25, 2013.

[118] R. Steeb, S. Cammarata, F. A. Hayes-Roth, P. W. Thorndyke, and R. E. Wesson, "Distributed intelligence for air fleet control," tech. rep., RAND CORP SANTA MONICA CA, 1981.

[119] E. Ephrati, J. S. Rosenschein, *et al.*, "Multi-agent planning as a dynamic search for social consensus," in *IJCAI*, vol. 93, pp. 423–429, 1993.

[120] M. R. Adler, A. B. Davis, R. Weihmayer, and R. W. Worrest, "Conflict-resolution strategies for nonhierarchical distributed agents," in *Distributed artificial intelligence*, pp. 139–161, Elsevier, 1998.

[121] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein, "Cooperation without communication," in *Readings in distributed artificial Intelligence*, pp. 220–226, Elsevier, 1988.

[122] L. S. E. Seow, S. K. Verma, Y. M. Mok, S. Kumar, S. Chang, P. Satghare, A. Hombali, J. Vaingankar, S. A. Chong, and M. Subramaniam, "Evaluating dsm-5 insomnia disorder and the treatment of sleep problems in a psychiatric population," *Journal of Clinical Sleep Medicine*, vol. 14, no. 02, pp. 237–244, 2018.

[123] V. Lux, S. Aggen, and K. Kendler, "The dsm-iv definition of severity of major depression: inter-relationship and validity," *Psychological medicine*, vol. 40, no. 10, pp. 1691–1701, 2010.

[124] S. S. Rika, "A proposal of a fully automated mental health care application for depression and insomnia," master's internship report, Utrecht University, 2018.

[125] A. P. Association *et al.*, *Diagnostic and statistical manual of mental disorders (DSM-5®).* American Psychiatric Pub, 2013.

[126] S. D. Hollon and P. C. Kendall, "Cognitive self-statements in depression: Development of an automatic thoughts questionnaire," *Cognitive therapy and research*, vol. 4, no. 4, pp. 383–395, 1980.

Appendix A

Knowledge Representation of SleepCare

This document contains a description of the various classes that are used for knowledge representation in the CoModel of InsomniaCoach modeled by [10]. We define the classes in bold letters and class objects in italic letters, for instance, **Trait** is a class and *trait* is an object.

The **CoModel** consists of 3 components:

- **Background**: The background knowledge of the coach. This knowledge does not change during the process.
- **CoProcess**: This is a (long term) repository for all the formal knowledge that is being built up during the coaching process.
- **CoActual** : This is a representation of the current state of the coachee as seen and maintained by the coachee.

A.1 Background

The background knowledge of the SleepCoach consists of:

A.1.1 disorders

List<Disorder>, a taxonomy of possible types of insomnia related conditions and their diagnostic characteristics. This information is used to enable the SleepCoach to check whether a consistent diagnostic picture emerges. There are various disorders that a coachee may actually have and every disorder has name, diagnostic description, and a probability score.

- *name* : String // All disorders are identifiable

- *description* : **Interaction_Recipe** //
- *probability* : **Score**

In insomnia, we might, for instance, distinguish the following disorders:

- Insomnia: sleep onset takes more than 30 minutes on majority of nights and sleep efficiency < 85 % (typically fatigue but not sleepy)
- Normal Aging (difficult to distinguish from primary insomnia)
- SBD (sleep related breathing disorder)
- PLMS RLS (restless Limb syndrome)
- Circadian Rhythm Sleep disorder
- Narcolepsy
- Parasomnias

A.1.2 traits

List<Trait>, a taxonomy of relevant properties of the coachee and the general situation of the coachee. The relevant traits include probable disorder, current progress in therapy, life circumstances, personal history, sleep characteristics, signed contracts, adherence etc. that help the coach to represent the coachee and to personalize utterances and distinguish cases that need to be handled differently. These are specified by names like "Sleep efficiency" and may have computational definitions. Traits are in fact a template (constructor) for the **Image**. It is thus similar to the **Image** in CoProcess, but may contain various default values.

- *signed contracts* (and their scores!!!) // initial default : empty
- *shared knowledge* // initial default : empty
- *sleep characterization* // initial default : normal sleep values
- effort/adherence/performance/motivation: Score //initial 0
- pre intake *insomnia history* of coachee (represented by some highlights or recorded question answers.) // initial default : empty
- *diagnosis probabilities* // initial default : population probabilities.

A.1.3 interaction recipes

List<Interaction_Recipe>, a collection of interaction recipes that can be called to perform a task that normally needs to happen only once and within a restricted timeframe, e.g. introduce the coach or the coachee, ask (a couple of) simple questions, perform (a stage of) intake, introduce the tools, or Technique. **Interaction_Recipe**s have clear goals and need to be «Playable». They may contain choices but are not very flexible in terms of timing, (as they are supposed to be handled during one interactive session, and they are supposed to be finished in one continuous timeblock) and they mainly effect straightforward knowledge transfer. **Interaction_Recipe**s can be played, and will result in updates of the **CoModel**.

A.1.4 interventions

List<Technique>,a collection of specifications/templates of therapy related techniques that coach may initiate and the coachee may need to adhere to, like filling in the sleepdiary, doing daily relaxation exercises, maintaining sleep restrictions, etc. These are the (generic) specifications of recurrent activity patterns that may shape habits and may well demand a certain discipline and motivation /commitment in order to be completed. This means that they need to be introduced to the coachee. The coachee may need to pass through several stages {precontemplation, contemplation, preparation} before she can commit to the Technique. These Techniques are repetitive and their specifications allow a more complicate timing structure as they specify ongoing cyclic patterns of activities that can show some (often quite small) variations from cycle to cycle. Some of their timing structure is specified through constraints. The **Technique** will contain feedback and support Interaction_Recipes that are «Playable» but, once a **Technique** is active, it may also be «Adjustable» to deal with the variations between the cycles. In such cases, it is probably best if we have a Tool that is associated to this Technique to streamline communication between coach and coachee about the desired adjustments.

- *name* : String;
- *appl* : {undetermined, nonapplicable, applicable} // see applicability constraint below.
- *stage* : { precontemplation, contemplation, preparation, action, maintained, terminated} // the stage determines the extent to which the coachee is involved in the technique.
- *intro* : **Interaction_Recipe** // an introduction interaction recipe includes explanations of the rationale behind the treatments, motivational information, and basic explanations about what to do and what to expect.

- *commitments* : List<**Contract**> //commitments of coachee needed for this Technique
- *tools* : List<**Tool**> // the tools that support the Technique
- *eval* : **Interaction_Recipe** //
- *motivation* : **Interaction_Recipe**
- *applicability* : List<**Constraint**> //condition of applicability of these **Technique**s which may be dependent on insomnia type, coachee health, coaching process progress or coachee situation, coachee knowledge and earlier treatment history.
- *order* : List<**Constraint**> //Constraints that specify order information for **Interaction_Recipe**s related to this Technique.
- *execute* : List<**Constraint**> //Any extra constraints to which the coach and coachee needs to adhere during the execution of a Technique.

There are various techniques, that need to be specified that all inherit from this general scheme.

- **SleepRestriction**:>**Technique**
- **FillOut Diary**:>**Technique**
- **Relaxation** :> **Technique**
- **CognitiveTherapy** :>**Technique**

The techniques may have a number of adjustable parameters. These parameters are normally specific for a given technique. Thus, a class like **SleepRestriction**, for instance, will have extra selectors corresponding to parameters relevant to sleep restriction, like a *threshold* time, and a *rising* time. The **Contract**s to which a user commits that play a role in this technique will refer to these values. Some of the parameters will be applicable to very specific contracts.

A.1.5 tools

List<**Tool**>, a collection of tools that are offered to the coachee, for instance the sleep diary, scheduling, relaxation tools, together with demonstration **Interaction_Recipe**s on how to introduce these tools and interpret the data that the coach may gain from them. All tools are «Touchable», the user can manipulate them, and the *dialogue_action*s with the tool lead to changes in the CoModel. The Tools often directly support specific Techniques.

The tools are structured information transfer tools, like the **sleep diary**, or the **Schedule**, or the **relax** tool, or perhaps a **sleep restriction** Tool. These tools can and will be used in the interventions, and it needs to be clear how information transferred in the tools affects the **CoModel**. Tools are «Touchable», «Identifiable» and «Callable»

- **Tool** <: **SleepDiary**
- **Tool** <: **Briefing**
- **Tool** <: **Relaxator**
- **Tool** <: **Schedule**
- **Tool** <: **SleepScheduler**
- **Tool** <: **Situation**

Each tool has:

- *name* : **String** // All Tools are «Identifiable»
- *seen* : **Boolean** // the tool has been introduced
- *used* : **Boolean** // the tool has been used
- *familiar* : **Boolean** //the tool has been understood
- *Method* : Call(**Tool**<*parameterlist*>) // runs the tool

A.1.6 templates

List<**Assignment**>, a repository of possible **Assignment**s that the coach may prescribe to the coachee. **Assignment**s are activities that the coachee needs to perform in the context of a certain technique, and that appear in entries in the **Schedule**, where the **Assignment**s are related to specific time blocks. The coachee commits to "performing" the scheduled **Assignment**s. There are various kinds of assignments: filling in the sleepdiary, doing a relaxation exercise, taking part in a briefing, or taking part in a cognitive therapy setting. Apart from a special kind of briefing (perhaps supported by a tool) to adjust the various parameters of the sleep schedule, there are no direct assignments for sleep scheduling. Sleep scheduling leads to many strong commitments, but sleeping or waking are not regarded as assignments, and the only assignments related to sleep scheduling are briefing sessions during which the sleep scheduling is set up, and (perhaps) cognitive therapy sessions during which misunderstandings about sleep therapy are cleared up. Every assignment has an evaluation which is supposed to take place at the end of the assignment. **Assignment**s might all be associated with a **Tool** that helps to execute them. An **Assignment** that is fully instantiated also needs to fit as an **Entry** in the database.

- *eval* : **Interaction_Recipe** // Every assignment will have its own kind of evaluation procedure. If a tool is used this need not involve further interaction with the coachee, e.g. to see that the sleep diary has been filled it may be enough to check a constraint, that may be defined through an attribute of the Tool.
- *start* : **Interaction_Recipe** // Every assignment may need a certain kick-off, like a reminder or the calling of an appropriate Tool.
- *description* : **Interaction_Recipe** // can explain what the assignment is to the coachee.
- *where* : **Interaction_Recipe** // can explain what optimal conditions for the assignment are like.
- *task* : Assigned // indicates what is supposed to happen in terms of Schedule
- *entry* : **Entry** // When is this assignment supposed to happen
- *status* : {unknown, planned, actual, cancelled}

FORALL A : Assignment HOLDS A.entry.proposition = A.task
There are various "subtypes" for assignments that have their own task, and evaluation procedure and kick-off.

- **SleepDiaryFill** :> **Assignment** // task = sleepdiary
- **LongBriefing** :> **Assignment** // task = briefing
- **RelaxationExercise** :> **Assignment** // task = relaxationExc
- **CogTherapySession** :> **Assignment** // task = cognitiveTherapy

A.1.7 Constraint

A **Constraint** is a condition that is supposed to apply to the coaching process. If the constraint dos not hold a conflict arises.

- *name* : **String** // All Constraints are «Identifiable»
- *description* : **Interaction_Recipe**
- *condition* : **Condition**

A.1.7.1 Behavioural Constraints

List <**Constraint**>, a collection of behavioural constraints. **Constraint**s are prescriptive rules that need to apply for the coaching process to be successful. For instance some constraints may impose a (partial) order on the **Interaction_Recipe**s that the coach is able to play. Other constraints are often associated with certain Techniques, and come into force when these Techniques are active. These constraints are mostly useful constraints on the coaching process that can be used to check a schedule or a situation for sources of possible problems, here known as Conflicts. A conflict is a constraint that is not satisfied. The InsomniaCoach can try to resolve conflicts in different ways, for instance, by asking the coachee to accept a proposed change in the schedule, (the preferred option, which then may or may not happen) or by downgrading "effort" and/or "adherence" of the coachee on a certain contract, which will keep the CoModel consistent, but produces less desirable results. If behavioural Constraints are changed, the behaviour of the coach is changing. (coaching parameters)

A.1.7.2 Informational Constraints

List <**Constraint**>, a set of informational constraints. For instance, if the InsomniaCoach is surprised, he needs to inform the coachee. Or if the coach needs actual information on sleep performance for at least five days a week, he needs to inform the coachee. Changing informational constraints will also change the behaviour of the coach. Many informational constraints do not need to be specified because they are implicit in the ordering constraints of Interaction_Recipes and Techniques.

A.2 CoProccess

The **CoProcess** is mainly a repository of all official historical and planning information about the coaching process, this contains all the information about the actual case that has been gathered by the coach to steer the ongoing process. This contains an **Image** of the Coachee as seen by the InsomniaCoach. It contains:

A.2.1 Image

This is an extensive model of the current state and development of the coachee, that is continuously updated. At each point in time, the coach maintains, as part of the **CoModel**, inside the **CoProcess** a particular **Image** of the coachee, which is initially filled with default values, and gets updated as certain traits (as given in the background) are being filled in within a particular coaching process for a particular coachee. So at each point on the timeline a particular Image exists. **Image** represents an abstraction of the coachee as maintained by the coach, which accumulates information about the coachee over time. It contains information about the actual state of diagnosis (averaged) sleep performance, and records the knowledge that has been shared with the coachee, the agreements that have been made with the coachee, and stores information that has been gained during introduction and intake sessions.

A.2.1.1 commitments

List<**Contract**>.A contract is a behavioural constraint, that acts like a signed contract, that a coachee may commit to. There may be various possible contracts that are found within the various techniques. These need to be further specified inside the techniques.

- *name* : **String** // All Contracts are «Identifiable»
- *description* : **Interaction_Recipe** condition : Condition
- *motivate* : **Interaction_Recipe**

A.2.1.2 seen

List<**Name**>, these are names of **Interaction_Recipe**s, **Fact**s, **Tool**s, **Score**s, **Technique**s, **Condition**s and **Contract**s, and represent what the coachee is supposed to know now (shared knowledge with InsomniaCoach), this may help the InsomniaCoach to determine what information the coachee already has, or what still needs to be conveyed. Of course, the coachee may sometimes decide to repeat information that it already conveyed.

A.2.1.3 sleep

Sleep is an object that represents how the coach currently characterises the sleep of the coachee. This might play a role in diagnosis, helps to make estimates of the user schedule, and plays an important role in sleep scheduling.

- *midsleep* : Daytime // estimation of moment in night when half the sleep time has passed (10 days)
- *ast* : Duration //average sleeping time (10 days)
- *gpt* : Daytime // average getPillowTime (10 days)
- *sot* : Duration //sleep onset time (10 days)
- *sef* : **Score** //sleep efficiency (10 days)
- *sas* : **Score** // sleep association score (10 days)

- *sfg* : **Score** //sleepfragmentation score (10 days)
- *ssa* : **Score** //subjective sleep score accuracy
- *awt* : Duration // average waking time in bed (10 days)
- *art* : Daytime // average rising time (10 days)
- *ibt* : Duration // in bed time

A.2.1.4 zeal

List<**Score**> scores of effort/adherence/performance/motivation per **Contract** or contract clause which may determine whether the coach proceeds or what it may emphasize or where it may try to motivate. **Score**s are performance or intensity indicators. They have a name and a description, a minimum and maximum value, an actual value, and they may have an optimal value.

- *name* : **String** // All Scores are «Identifiable»
- *description* : **Interaction_Recipe** // Explains the meaning of the Score.
- *currentValue* : **Real** // a number, the (last known) measurement for the score.
- *minimum* : **Real** // minimum value
- *maximum* : **Real** // maximum value
- *optimisable* : **Boolean** // whether this score has an optimum.
- *optimum* : **Real** // optimal value (optional)

A.2.1.5 facts

List<**Fact**> are insomnia history (represented by some highlights or recorded question answers. This may contain information excerpts from the interviews that is useful to influence motivation. The idea is that question answer pairs during interviews can be compiled into facts that can be referred to later.

A problem for the coach is that, in order to increase motivation, he may need handles "handles" to be able to refer back to personal information from the coachee that has been obtained during the intake. For instance, a coachee may have identified a significant life event that acted as a "precipitating event" when insomnia problems started. Or the coachee may have identified a "motivating goal" that stands behind her decision to seriously engage her sleeping problems. If one would later like to refer to the "precipitating event" or to the "motivating goal" one does, of course, want to refer back to the concrete and meaningful event or goal that has been provided by the coachee herself. One way to do this may be to "memorise" or "make a snapshot" of the answer of the coachee to particular questions during introduction or intake, so that these can be played back later. Alternatively, one may explicitly ask the coachee to make a drawing or a photo that reminds her of this goal or event. The resulting handle is called a "**Fact**" and might contain:

- *role* :{ precipitating event, motivating goal, ... }
- *recording* : «Playable»

A.2.1.6 diagnosis

probabilities that may determine applicability of a technique e.g. whether the coach can proceed with it or which next steps to take)

A.2.2 techniques

List<**Technique**>, a collection contains all the information about existing Techniques for the given coachee, so it registers the applicability and keeps track of their development in terms of stages {precontemplation, contemplation, preparation, action, maintenance, termination} and also keeps track of all the essential parameters of the **Technique**. (like for the case of **SleepRestriction**: *thresholdTime*, *RisingTime* etc. which may be adjusted over time).

A.2.3 Schedule

A schedule is essentially a collection of entries and events. There will be a corresponding **Schedule** tool that renders a schedule «Touchable». So we can select subobjects of the **Schedule** like Entries or Events. Entries have a start and stop Event and relate to TimeBlocks. All Entries and Events relate to a specific Proposition.

- *entryList* : List<**Entry**>
- *eventList* : List<**Event**>
- *getFocusEntry* : **Entry** // selected Entry
- *getFocusEvent* : **Event** // selected Event
- *now* : TimePoint // always points to the current moment

In principle, one can create "subschedules" that are only concerned with specific Propositions or a specific TimeBlock by appying filters to a larger Schedule. In particular, one can filter entries over a given (large) timeblock. Schedules allow us to compute estimates of the value of propositions.

- *estValue*(p : **Proposition**, t : Timepoint) : Boolean
- *estStatus*(p : **Proposition**, t : Timepoint) : {default, planned, actual}

Returns a Boolean and status. Computes an estimation of the value for the proposition argument that is likely to hold immediately after the timepoint argument. The estimation is based on the stretch of time before and at the timepoint. If no such stretch of time exists, (e.g. the timepoint corresponds to the start of the schedule) a default value is returned. The computation uses the last (non-cancelled) event on or before the timepoint that pertains to this proposition, this may be a probing event or a switchevent. Reliability of the estimate (as roughly indicated by the status, which is derived from the status of the event used) may of course vary, but will in general be greater for past timepoints then for future timepoints.

- *curValue*(p: **Proposition**) = estvalue(p : **Proposition**, now)
- *curStatus*(p: **Proposition**) = estStatus(p : **Proposition**, now)

For the sleep domain, the coach will always maintain the following subschedules:

I *today* : **Schedule** // a schedule corresponding to a 24 hours block from midsleep+12 till midsleep + 12 that contains the current moment.

II *tonight* : **Schedule** // a schedule corresponding to a 24 hours block from midsleep till midsleep that contains the current moment.

INVARIANT that is maintained:

- today.start < now < today.stop
- tonight.start < now < tonight.stop

Note: midsleep is a Daytime, so these invariants imply that the Schedules need updating every 12 hours. Practically updating is only needed when coach or coachee wants to access some Schedule. For synchronisation and buffering purposes and the like it may be useful to also have Schedules for "yesterday", "tomorrow" "last night" and "tomorrow night".

A.2.3.1 Entry

Each **Entry** has a TimeBlock, and a **Proposition** that indicates what is true during the *TimeBlock* corresponding to the **Entry**. Entries have a *start* event and a *stop* event. (These events are always **SwitchEvent**s). Entry specifies in which *TimeBlock* the **Occupation** is supposed to take place.

- *start* : **SwitchEvent**

- *stop* : **SwitchEvent**
- *what* : **Proposition**
- *block* : *TimeBlock*
- *status* : {unknown,planned, actual, cancelled} //(= start.status)

Invariant:

- FORALL e : **Entry** HOLDS e.start.time = e.block.start AND e.start.onOff = on
- FORALL e : **Entry** HOLDS e.stop.time = e.block.stop AND e.stop.onOff = off
- FORALL e : **Entry** HOLDS e.occup = e.start.**Proposition** = e.stop.**Proposition**

A.2.3.2 TimeBlock

Basic single block of time, continuous from start to stop.

- *start* : TimePoint
- *stop* : TimePoint
- *length* : Duration

A.2.3.3 TimePoint

This is a primitive data type. Time points are uniquely identifiable moments in "time".

A.2.3.4 Duration

This is a primitive data type. Durations can be calculated from the "difference" between two uniquely identifiable moments in "time". Durations can be negative or positive.

A.2.3.5 Daytime

Daytime is a primitive datatype that is used to point to a particular point in the 24 hour day. Intuitively, it has values between 00.00 and 23.59.59. Daytime can be meaningfully compared with ordinary TimePoints, whenever comparison takes place within 24 hours TimeBlock. (As within such a TimeBlock, the daytime always corresponds to a unique TimePoint.)

A.2.3.6 Event

All **Event**s have a TimePoint at which they will take place, and they have a status. The status of an event (and all its associated information) within the **Schedule** is planned, canceled or actual. The actual **Event** may have another (actual) TimePoint as originally planned. The status "planned' is also used for estimates. **Event**s can change or determine the value of Propositions.

- *time* : TimePoint
- *status* : {unknown,planned, actual, cancelled }
- *proposition* : **Proposition** //the proposition being switched or measured
- **Event** <: **SwitchEvent**
- **Event** <: **ProbeEvent**
- **Event** <: **SingularEvent**

A.2.3.7 Proposition

Propositions specify what can be true or false about the coachee; the coachee may be in a certain situation and busy with a certain occupation. The value of a proposition depends on **Event**s and can vary over time, so one can only determine whether a **Proposition** is true or false before or after certain Events. These Events are recorded in the **Schedule**, and one therefore needs to turn to the **Schedule** to get this information.

- *name* : **String** // Propositions are «Identifiable»
- *description* : **Interaction_Recipe** // describes what the proposition means

We identify two main types of propositions:

I **Proposition** <: **Occupation**

II **Proposition** <: **Circumstance**

A.2.3.8 Occupation

Occupations are mutual exclusive activities that keep you busy while you do them. Thus **Occupation**s cannot overlap. This means that any event that switches on an **Occupation**, switches off any other **Occupation** that is still ongoing. Entries relating to occupations can be found in the **userSchedule**.

- **Occupation** <: **Shared**
- **Occupation** <: **Assigned**

A.2.3.9 SwitchEvent

This is an event that switches a proposition on or off. These always come in pairs and will naturally define TimeBlocks.

- *proposition* : **Proposition** //the proposition being switched
- *onOff* : {on , off} // on switches from FALSE to TRUE // off switches from TRUE to FALSE

A.2.3.10 ProbeEvent

This is a "measurement" type of event that determines whether a **Propostion** is true or false at a certain point in time. These **Event**s can occur due to sensors or because the coache herself presents information in **CoActual**. Actually performing the measurement may either generate the event or change the status of a planned **ProbeEvent**.

- *value* :**Boolean** // FALSE or TRUE

A.2.3.11 SingularEvent

There may be singular event that changes the state of the coachee but do not have a clear counterpart. An example is taking a dose of caffeine through coffee or chocolate, etc. We have included these to signal that they may need to be considered future. **SingularEvent**s may, in the future, probably be meaningfully related to **Score**s. Currently **SingularEvent**s may only enable us to specify simple **Constraint**s. Note that it may be possible to model a simple **Constraint** for a case like caffeine intake just as well with the help of switch events. (However, this may be unable to model the effect double doses). Currently, the class **SingularEvent** inherits from **Event** and has no additional selectors. An example of a singular Event might be;

- *caffeine* : **SingularEvent**

A.2.3.12 Circumstance

Circumstances one can be in. **Circumstance**s are **Proposition**s that can be true at the same time. E.g. one might be at home in bed watching TV in the bedroom, with the light on, which also is a Screen activity. The class **Circumstance** inherits from **Proposition** and has no additional selectors. We can, however, for the sleep domain, enumerate various circumstances as unique and identifiable inhabitants of the class Circumstance.

- *inBed* : **Circumstance**

- *atHome* : **Circumstance**
- *seeTV* : **Circumstance**
- *seeScreen* : **Circumstance**
- *inBedRoom* : **Circumstance**
- *lightHigh* : **Circumstance**
- *noisy* : **Circumstance**
- *inTravel* : **Circumstance**
- *atOffice* : **Circumstance**
- *awake* : **Circumstance**

A.2.3.13 Assigned

There are a number of **Assigned** activities/**Occupation**s that may be planned for the coachee by the coach. The class **Assigned** inherits from **Occupation** and has no additional selectors. We can, however, for the sleep domain, enumerate various shared occupations as unique and identifiable inhabitants of the class Assigned.

- **relaxationExc : Assigned**
- **sleepdiary : Assigned**
- **briefing : Assigned**
- **intake : Assigned**
- **cognitiveTherapy : Assigned**

A.2.3.14 Shared

There are a number of personal yet **Shared Occupation**s that the Insomnia-Coach may be informed about. The class **Shared** inherits from **Occupation** and has no additional selectors. We can, however, for the sleep domain, enumerate various shared occupations as unique and identifiable inhabitants of the class Shared.

- *eat* : **Shared**
- *play* : **Shared**
- *work* : **Shared**
- *sleep* : **Shared**

A.2.3.15 userSchedule

This is a time **Schedule** of planned **Assignment**s and **Shared** user **Occupation**s that can be seen and edited by the coachee. This also contains information on the **Occupation**s of the coachee that are **Shared**. We may be able to read in this **Schedule** when the coachee was watching TV, eating, awake, etc. So this time **Schedule** reflects facts and estimations of (changes in) the **Situation** of the coach at various points in time.

A.2.3.16 feelSchedule

This is a time **Schedule** that reflects the development of subjective **Feel** parameters in the past, and which may also contain projections that try to indicate how she might **Feel** in the (near) future.

A.2.3.17 dealSchedule

This is a time **Schedule** that reflects when certain **Contract**s where agreed to, or when they were retracted, and how they were adhered to in the mean time. All these **Schedule**s use essentially the same time parameter and can therefore be overlaid on one another. So one can also combine all the **Schedule**s in one large **Schedule** that allows different views. The implementation of **Schedules** is quite straightforward. They consist basically of lists of entries referring to **Shared Occupation**s or **Assignment**s that relate to blocks of time or events that relate to points in time.

A.3 CoActual

The **CoActual** class contains the following information:

A.3.1 feel

List<**Feel**>, a collection of personal subjective "feeling" **Score**s like *stress, general wellbeing, subjective sleeping experience or fitness*. These parameters may be important for the coach to get some feedback on the effect of the therapy, and also may enable the coach to act empathically when the coachee is having a hard time. The **Feel** may be supported by a **Tool** that makes these parameters «Touchable».

- *stress* : **Score** //How stressed you are
- *sleepQuality* : **Score** //Perceived sleep quality
- *wellbeing* : **Score** // General wellbeing

- *fitness* : **Score** // How fit do you feel?

A.3.2 sit

List<**Situation**>, a collection of personal objective information concerning location, awake, eating, etc. The **Situation** represents aspects of the current **Situation** of the coachee that the coachee itself can easily and reliably judge or estimate, e.g. **Circumstance**s like *being awake, in bed or not, watching TV*, personal **Occupation**s like *eating* or *working*, as well as activities assigned by the InsomniaCoach like and timing estimates like at what time she hit the pillow or rose in the morning. The **Situation** may be supported by a **Tool** that allows the coachee to confirm, correct, or update the relevant values. Some of these values relevant to the sleep domain are indicated below:

- *inBed* : **Circumstance**
- *atHome* : **Circumstance**
- *seeTV* : **Circumstance**
- *seeScreen* : **Circumstance**
- *inBedRoom* : **Circumstance**
- *lightHigh* : **Circumstance**
- *noisy* : **Circumstance**
- *inTravel* : **Circumstance**
- *atOffice* : **Circumstance**
- *awake* : **Circumstance**
- *eat* : **Shared**
- *play* : **Shared**
- *work* : **Shared**
- *sleep* : **Shared** // might be signalled by hardware/actigraphy.
- *getPillowTime* : Daytime // estimation of time when you close your eyes to sleep (maybe unknown)
- *getRisingTime* : Daytime // estimation of time when you get up (maybe unknown)

A.3.3 tools

List<**Tool**>, a repertoire of tools that the coachee knows how to use.

A.3.4 identity

The **Identity** class contains *name*, *birth date, preference*s etc. this might also contain further info, like *profession.*

A.3.5 seen

List<**Name**>, the various concepts, **Technique**s, **Fact**s, **Interaction_Recipe**s the coachee is already familiar with.

A.3.6 deal

List<**Contract**>, a list of all actual agreements between the coachee and the coach

A.3.7 doubt

List<**Contract**>, a list of agreements proposed or promoted by the coach that the coachee has not yet agreed to.

Appendix B

Knowledge Representation of DepressionCare

This document contains a description of the various classes that are used for knowledge representation in the CoModel of DepressionCoach modeled by [124]. We define the classes in bold letters and class objects in italic letters, for instance, **Trait** is a class and *trait* is an object.

The **CoModel** consists of 3 components:

- **Background**: The background knowledge of the coach. This knowledge does not change during the process.
- **CoProcess**: This is a (long term) repository for all the formal knowledge that is being built up during the coaching process.
- **CoActual** : This is a representation of the current state of the coachee as seen and maintained by the coachee.

B.1 Background

The background knowledge of the SleepCoach consists of:

B.1.1 disorders

List<Disorder>, a taxonomy of possible types of depression related conditions and their diagnostic characteristics. This information is used to enable the DepressionCoach to check whether a consistent diagnostic picture emerges. A Disorder can be any psychological or somatic symptoms that a coachee may have in depression. Disorders have name, a diagnostic description, and a probability score.

- name : String
  All disorders are identifiable
- description : Interaction_Recipe
- probability : Score

We distinguish the following disorders according to DSM- IV [125] and each of them is scored based on how often the coachee feel or experience it over the last two weeks

- name : Loss of Interest or Pleasure
  Description : The coachee feels loss of interest or pleasure in doing daily activities
- name: Depressed mood
  Description : The coachee feels sad, empty or down.
- name : Change in Sleep
  Description : Insomnia or Hypersomnia
- name: Fatigue or Loss of energy
  Description : The coachee feels tired or less energetic while doing daily activities.
- name: Change in Appetite
  Description : Poor appetite or overeating or significant weight change (around 5%)
- name : Feelings of Worthlessness or Guilt
  Description : The coachee feels bad about herself and holds herself responsible for everything wrong that is happening to her or her family
- name : Lack of Concentration
  Description : The coachee has trouble concentrating on things like reading the newspaper or watching TV.
- name : Psychomotor Agitation or Retardation
  Description : The coachee starts doing activities differently than the way she naturally does, i.e., moving too slowing or too fast or being too restless that other people can observe the change in herself.
- name : Suicidality
  Description : Thoughts of death or suicide.

Scoring is done as how often the coachee feels a disorder over the last two weeks :

0 :> not at all

1 :> several days

2 :> more than half the days

3 :> nearly everyday

The overall severity measure od depression is done as:

Severity :> Total score of all the symptoms / 27
Depression Severity :

0-4 : none

5-9 : mild

10-14 : moderate

15-19 : moderately severe

20-27 : severe

The severity measure and scoring process of DepressionCare are different from SleepCare system. The severity measure or the assessment of depression level is essential because the different levels of depression may require different types of therapy, for instance, severe level of depression is very vulnerable and may require intensive care of a human coach rather than an e-coach. Also to judge whether a coachee is suitable for the coaching process, the coach needs to assess the coachee's depression level, for instance, none and severe level coachees may not be fit for the coaching program.

B.1.2 traits

List<Trait>, a taxonomy of relevant properties of the coachee and the general situation of the coachee. The Traits are in fact a template (constructor) for the Image. It may contain various default values.

- signed contracts (and their Scores) // initial default : empty
- shared knowledge // initial default : empty
- mood characterization // initial default : normal sleep values
- thought characterization // depends on the ThoughtDiary and the initial value is empty. To identify automatic thoughts thought characterization is important, hence this is included in the DepressionCare.

- effort/adherence/performance/motivation scores // none, .
- pre intake : history of any kind of abuse (physical or mental), social and family environment of the coachee (represented by some highlights or recorded question answers.) // initial default : empty
- diagnosis probabilities // initial default : population probabilities.

B.1.3 automaticThoughts

List<**AutomaticThoughts**>, a list of 30 automatic thoughts (e.g., [126]). The coach can compare the thoughts of the coachee with this list to assess her thinking process. This class is new in DepressionCare and is used mainly for assessment.

B.1.4 interaction recipes

List<Interaction_Recipe>, a collection of interaction recipes that can be called to perform a task that normally needs to happen only once and within a restricted timeframe, e.g. introduce the coach or the coachee, ask (a couple of) simple questions, perform (a stage of) intake, introduce the tools, or Technique. **Interaction_Recipe**s have clear goals and need to be «Playable». They may contain choices but are not very flexible in terms of timing, (as they are supposed to be handled during one interactive session, and they are supposed to be finished in one continuous timeblock) and they mainly effect straightforward knowledge transfer. **Interaction_Recipe**s can be played, and will result in updates of the **CoModel**.

B.1.5 interventions

List<Technique>,a collection of specifications/templates of therapy related techniques that coach may initiate and the coachee may need to adhere to, like filling in the sleepdiary, doing daily relaxation exercises, maintaining sleep restrictions, etc. These are the (generic) specifications of recurrent activity patterns that may shape habits and may well demand a certain discipline and motivation /commitment in order to be completed. This means that they need to be introduced to the coachee. The coachee may need to pass through several stages {precontemplation, contemplation, preparation} before she can commit to the Technique. These Techniques are repetitive and their specifications allow a more complicate timing structure as they specify ongoing cyclic patterns of activities that can show some (often quite small) variations from cycle to cycle. Some of their timing structure is specified through constraints. The **Technique** will contain feedback and support Interaction_Recipes that are «Playable» but, once a **Technique** is active, it may also be «Adjustable» to deal with the variations between the cycles. In such cases, it is probably best if we have a Tool that is associated to this Technique to streamline communication between coach and coachee about the desired adjustments.

- *name* : String;
- *appl* : {undetermined, nonapplicable, applicable} // see applicability constraint below.
- *stage* : { precontemplation, contemplation, preparation, action, maintained, terminated} // the stage determines the extent to which the coachee is involved in the technique.
- *intro* : **Interaction_Recipe** // an introduction interaction recipe includes explanations of the rationale behind the treatments, motivational information, and basic explanations about what to do and what to expect.
- *commitments* : List<**Contract**> //commitments of coachee needed for this Technique
- *tools* : List<**Tool**> // the tools that support the Technique
- *eval* : **Interaction_Recipe** //
- *motivation* : **Interaction_Recipe**
- *applicability* : List<**Constraint**> //condition of applicability of these **Technique**s which may be dependent on insomnia type, coachee health, coaching process progress or coachee situation, coachee knowledge and earlier treatment history.
- *order* : List<**Constraint**> //Constraints that specify order information for **Interaction_Recipe**s related to this Technique.
- *execute* : List<**Constraint**> //Any extra constraints to which the coach and coachee needs to adhere during the execution of a Technique.

The DepressionCare has more techniques to use in the coaching process than the SleepCare system.

- Activity Scheduling :> Technique
- Cognitive Rehearsal :> Technique
- Self-Reliance :> Technique
- Role-Playing :> Technique
- Role-Reversal :> Technique

- Distraction :> Technique
- Recording Automatic Thoughts :> Technique
- Testing Automatic Thoughts :> Technique
- Reattribution :> Technique
- Reconceptualization of Problems/ Situations :> Technique

The techniques may have a number of adjustable parameters. These parameters are normally specific for a given technique. Thus, a class like **SleepRestriction**, for instance, will have extra selectors corresponding to parameters relevant to sleep restriction, like a *threshold* time, and a *rising* time. The **Contract**s to which a user commits that play a role in this technique will refer to these values. Some of the parameters will be applicable to very specific contracts.

B.1.6 tools

List<**Tool**>, a collection of tools that are offered to the coachee, for instance the sleep diary, scheduling, relaxation tools, together with demonstration **Interaction_Recipe**s on how to introduce these tools and interpret the data that the coach may gain from them. All tools are «Touchable», the user can manipulate them, and the *dialogue_action*s with the tool lead to changes in the CoModel. The Tools often directly support specific Techniques.

The tools are structured information transfer tools, like the **sleep diary**, or the **Schedule**, or the **relax** tool, or perhaps a **sleep restriction** Tool. These tools can and will be used in the interventions, and it needs to be clear how information transferred in the tools affects the **CoModel**. Tools are «Touchable», «Identifiable» and «Callable»

- Tool <: ActivityDiary
- Tool <: Briefing
- Tool <: ThoughtDiary
- Tool <: Schedule
- Tool <: Self-Help
- Tool <: Situation

Each tool has:

- *name* : **String** // All Tools are «Identifiable»
- *seen* : **Boolean** // the tool has been introduced

- *used* : **Boolean** // the tool has been used
- *familiar* : **Boolean** //the tool has been understood
- *Method* : Call(**Tool**<*parameterlist*>) // runs the tool

B.1.7 Assignment

List<**Assignment**>, a repository of possible **Assignment**s that the coach may prescribe to the coachee. **Assignment**s are activities that the coachee needs to perform in the context of a certain technique, and that appear in entries in the **Schedule**, where the **Assignment**s are related to specific time blocks. The coachee commits to "performing" the scheduled **Assignment**s. There are various kinds of assignments: filling in the sleepdiary, doing a relaxation exercise, taking part in a briefing, or taking part in a cognitive therapy setting. Apart from a special kind of briefing (perhaps supported by a tool) to adjust the various parameters of the sleep schedule, there are no direct assignments for sleep scheduling. Sleep scheduling leads to many strong commitments, but sleeping or waking are not regarded as assignments, and the only assignments related to sleep scheduling are briefing sessions during which the sleep scheduling is set up, and (perhaps) cognitive therapy sessions during which misunderstandings about sleep therapy are cleared up. Every assignment has an evaluation which is supposed to take place at the end of the assignment. **Assignment**s might all be associated with a **Tool** that helps to execute them. An **Assignment** that is fully instantiated also needs to fit as an **Entry** in the database.

- *eval* : **Interaction_Recipe** // Every assignment will have its own kind of evaluation procedure. If a tool is used this need not involve further interaction with the coachee, e.g. to see that the sleep diary has been filled it may be enough to check a constraint, that may be defined through an attribute of the Tool.
- *start* : **Interaction_Recipe** // Every assignment may need a certain kick-off, like a reminder or the calling of an appropriate Tool.
- *description* : **Interaction_Recipe** // can explain what the assignment is to the coachee.
- *where* : **Interaction_Recipe** // can explain what optimal conditions for the assignment are like.
- *task* : Assigned // indicates what is supposed to happen in terms of Schedule
- *entry* : **Entry** // When is this assignment supposed to happen

- *status* : {unknown, planned, actual, cancelled}

FORALL A : Assignment HOLDS A.entry.proposition = A.task
There are various "subtypes" for assignments that have their own task, and evaluation procedure and kick-off.

- ThoughtRecord :> Assignment
  task = filling in automatic thoughts in the "Thought" column of the Thought-Diary

- SituationRecord :> Assignment
  task = filling in the relevant situations or events that stimulate the particular thought

- HypothesisRecord :> Assignment
  task = recording hypothesis or rationale behind the particular thought

- EvidenceRecord :> Assignment
  task = recording the evidences that support a hypothesis

- TestingHypothesis :> Assignment
  task = testing the hypothesis with the available evidences

- FindAlternativeThought :> Assignment
  task= finding alternative thoughts for the particular situation or event.

- Judgment :> Assignment
  task = deciding whether the thought is justified or not?

B.1.8 Constraint

A **Constraint** is a condition that is supposed to apply to the coaching process. If the constraint dos not hold a conflict arises.

- *name* : **String** // All Constraints are «Identifiable»

- *description* : **Interaction_Recipe**

- *condition* : **Condition**

B.1.8.1 Behavioural Constraints

List <**Constraint**>, a collection of behavioural constraints. **Constraint**s are prescriptive rules that need to apply for the coaching process to be successful. For instance some constraints may impose a (partial) order on the **Interaction_Recipe**s that the coach is able to play. Other constraints are often associated with certain Techniques, and come into force when these Techniques are active. These constraints are mostly useful constraints on the coaching process that can be used to check a schedule or a situation for sources of possible problems, here known as Conflicts. A conflict is a constraint that is not satisfied. The InsomniaCoach can try to resolve conflicts in different ways, for instance, by asking the coachee to accept a proposed change in the schedule, (the preferred option, which then may or may not happen) or by downgrading "effort" and/or "adherence" of the coachee on a certain contract, which will keep the CoModel consistent, but produces less desirable results. If behavioural Constraints are changed, the behaviour of the coach is changing. (coaching parameters)

B.1.8.2 Informational Constraints

List <**Constraint**>, a set of informational constraints. For instance, if the InsomniaCoach is surprised, he needs to inform the coachee. Or if the coach needs actual information on sleep performance for at least five days a week, he needs to inform the coachee. Changing informational constraints will also change the behaviour of the coach. Many informational constraints do not need to be specified because they are implicit in the ordering constraints of Interaction_Recipes and Techniques.

B.2 CoProccess

The **CoProcess** is mainly a repository of all official historical and planning information about the coaching process, this contains all the information about the actual case that has been gathered by the coach to steer the ongoing process. This contains an **Image** of the Coachee as seen by the InsomniaCoach. It contains:

B.2.1 Image

This is an extensive model of the current state and development of the coachee, that is continuously updated. At each point in time, the coach maintains, as part of the **CoModel**, inside the **CoProcess** a particular **Image** of the coachee, which is initially filled with default values, and gets updated as certain traits (as given in the background) are being filled in within a particular coaching process for a particular coachee. So at each point on the timeline a particular Image exists. **Image** represents an abstraction of the coachee as maintained by the coach, which accumulates information about the coachee over time. It contains information about the actual state of diagnosis (averaged) sleep performance, and records the knowledge that has been shared with the coachee, the agreements that have been made with the coachee, and stores information that has been gained during introduction and intake sessions.

B.2.1.1 commitments

List<**Contract**>.A contract is a behavioural constraint, that acts like a signed contract, that a coachee may commit to. There may be various possible contracts that are found within the various techniques. These need to be further specified inside the techniques.

- *name* : **String** // All Contracts are «Identifiable»
- *description* : **Interaction_Recipe** condition : Condition
- *motivate* : **Interaction_Recipe**

B.2.1.2 seen

List<**Name**>, these are names of **Interaction_Recipe**s, **Fact**s, **Tool**s, **Score**s, **Technique**s, **Condition**s and **Contract**s, and represent what the coachee is supposed to know now (shared knowledge with InsomniaCoach), this may help the InsomniaCoach to determine what information the coachee already has, or what still needs to be conveyed. Of course, the coachee may sometimes decide to repeat information that it already conveyed.

B.2.1.3 thought

List<**Thought**> represents how the coach currently characterises the thoughts of the coachee.

B.2.1.4 zeal

List<**Score**> scores of effort/adherence/performance/motivation per **Contract** or contract clause which may determine whether the coach proceeds or what it may emphasize or where it may try to motivate. **Score**s are performance or intensity indicators. They have a name and a description, a minimum and maximum value, an actual value, and they may have an optimal value.

- *name* : **String** // All Scores are «Identifiable»
- *description* : **Interaction_Recipe** // Explains the meaning of the Score.
- *currentValue* : **Real** // a number, the (last known) measurement for the score.
- *minimum* : **Real** // minimum value
- *maximum* : **Real** // maximum value
- *optimisable* : **Boolean** // whether this score has an optimum.
- *optimum* : **Real** // optimal value (optional)

B.2.1.5 facts

List<**Fact**> are insomnia history (represented by some highlights or recorded question answers. This may contain information excerpts from the interviews that is useful to influence motivation. The idea is that question answer pairs during interviews can be compiled into facts that can be referred to later.

A problem for the coach is that, in order to increase motivation, he may need handles "handles" to be able to refer back to personal information from the coachee that has been obtained during the intake. For instance, a coachee may have identified a significant life event that acted as a "precipitating event" when insomnia problems started. Or the coachee may have identified a "motivating goal" that stands behind her decision to seriously engage her sleeping problems. If one would later like to refer to the "precipitating event" or to the "motivating goal" one does, of course, want to refer back to the concrete and meaningful event or goal that has been provided by the coachee herself. One way to do this may be to "memorise" or "make a snapshot" of the answer of the coachee to particular questions during introduction or intake, so that these can be played back later. Alternatively, one may explicitly ask the coachee to make a drawing or a photo that reminds her of this goal or event. The resulting handle is called a "**Fact**" and might contain:

- *role* :{ precipitating event, motivating goal, ... }
- *recording* : «Playable»

B.2.1.6 diagnosis

probabilities that may determine applicability of a technique e.g. whether the coach can proceed with it or which next steps to take)

B.2.2 techniques

List<**Technique**>, a collection contains all the information about existing Techniques for the given coachee, so it registers the applicability and keeps track of their development in terms of stages {precontemplation, contemplation, preparation, action, maintenance, termination} and also keeps track of all the essential parameters of the **Technique**. (like for the case of **SleepRestriction**: *thresholdTime*, *RisingTime* etc. which may be adjusted over time).

B.2.3 Schedule

A schedule is essentially a collection of entries and events. There will be a corresponding **Schedule** tool that renders a schedule «Touchable». So we can select subobjects of the **Schedule** like Entries or Events. Entries have a start and stop Event and relate to TimeBlocks. All Entries and Events relate to a specific Proposition.

- *entryList* : List<**Entry**>
- *eventList* : List<**Event**>
- *getFocusEntry* : **Entry** // selected Entry
- *getFocusEvent* : **Event** // selected Event
- *now* : TimePoint // always points to the current moment

In principle, one can create "subschedules" that are only concerned with specific Propositions or a specific TimeBlock by appying filters to a larger Schedule. In particular, one can filter entries over a given (large) timeblock. Schedules allow us to compute estimates of the value of propositions.

- *estValue*(p : **Proposition**, t : Timepoint) : Boolean
- *estStatus*(p : **Proposition**, t : Timepoint) : {default, planned, actual}

Returns a Boolean and status. Computes an estimation of the value for the proposition argument that is likely to hold immediately after the timepoint argument. The estimation is based on the stretch of time before and at the timepoint. If no such stretch of time exists, (e.g. the timepoint corresponds to the start of the schedule) a default value is returned. The computation uses the last (non-cancelled) event on or before the timepoint that pertains to this proposition, this may be a probing event or a switchevent. Reliability of the estimate (as roughly indicated by the status, which is derived from the status of the event used) may of course vary, but will in general be greater for past timepoints then for future timepoints.

- *curValue*(p: **Proposition**) = estvalue(p : **Proposition**, now)
- *curStatus*(p: **Proposition**) = estStatus(p : **Proposition**, now)

For the sleep domain, the coach will always maintain the following subschedules:

I *today* : **Schedule** // a schedule corresponding to a 24 hours block from midsleep+12 till midsleep + 12 that contains the current moment.

II *tonight* : **Schedule** // a schedule corresponding to a 24 hours block from midsleep till midsleep that contains the current moment.

INVARIANT that is maintained:

- today.start < now < today.stop

- tonight.start < now < tonight.stop

Note: midsleep is a Daytime, so these invariants imply that the Schedules need updating every 12 hours. Practically updating is only needed when coach or coachee wants to access some Schedule. For synchronisation and buffering purposes and the like it may be useful to also have Schedules for "yesterday", "tomorrow" "last night" and "tomorrow night".

B.2.3.1 Entry

Each **Entry** has a TimeBlock, and a **Proposition** that indicates what is true during the *TimeBlock* corresponding to the **Entry**. Entries have a *start* event and a *stop* event. (These events are always **SwitchEvent**s). Entry specifies in which *TimeBlock* the **Occupation** is supposed to take place.

- *start* : **SwitchEvent**
- *stop* : **SwitchEvent**
- *what* : **Proposition**
- *block* : *TimeBlock*
- *status* : {unknown,planned, actual, cancelled} //(= start.status)

Invariant:

- FORALL e : **Entry** HOLDS e.start.time = e.block.start AND e.start.onOff = on
- FORALL e : **Entry** HOLDS e.stop.time = e.block.stop AND e.stop.onOff = off
- FORALL e : **Entry** HOLDS e.occup = e.start.**Proposition** = e.stop.**Proposition**

B.2.3.2 TimeBlock

Basic single block of time, continuous from start to stop.

- *start* : TimePoint
- *stop* : TimePoint
- *length* : Duration

B.2.3.3 TimePoint

This is a primitive data type. Time points are uniquely identifiable moments in "time".

B.2.3.4 Duration

This is a primitive data type. Durations can be calculated from the "difference" between two uniquely identifiable moments in "time". Durations can be negative or positive.

B.2.3.5 Daytime

Daytime is a primitive datatype that is used to point to a particular point in the 24 hour day. Intuitively, it has values between 00.00 and 23.59.59. Daytime can be meaningfully compared with ordinary TimePoints, whenever comparison takes place within 24 hours TimeBlock. (As within such a TimeBlock, the daytime always corresponds to a unique TimePoint.)

B.2.3.6 Event

All **Event**s have a TimePoint at which they will take place, and they have a status. The status of an event (and all its associated information) within the **Schedule** is planned, canceled or actual. The actual **Event** may have another (actual) TimePoint as originally planned. The status "planned' is also used for estimates. **Event**s can change or determine the value of Propositions.

- *time* : TimePoint
- *status* : {unknown,planned, actual, cancelled }
- *proposition* : **Proposition** //the proposition being switched or measured
- **Event** <: **SwitchEvent**
- **Event** <: **ProbeEvent**
- **Event** <: **SingularEvent**

B.2.3.7 Proposition

Propositions specify what can be true or false about the coachee; the coachee may be in a certain situation and busy with a certain occupation. The value of a proposition depends on **Event**s and can vary over time, so one can only determine whether a **Proposition** is true or false before or after certain Events. These Events are recorded in the **Schedule**, and one therefore needs to turn to the **Schedule** to get this information.

- *name* : **String** // Propositions are «Identifiable»
- *description* : **Interaction_Recipe** // describes what the proposition means

We identify two main types of propositions:

I **Proposition** <: **Occupation**

II **Proposition** <: **Circumstance**

B.2.3.8 Occupation

Occupations are mutual exclusive activities that keep you busy while you do them. Thus **Occupation**s cannot overlap. This means that any event that switches on an **Occupation**, switches off any other **Occupation** that is still ongoing. Entries relating to occupations can be found in the **userSchedule**.

- **Occupation** <: **Shared**
- **Occupation** <: **Assigned**

B.2.3.9 SwitchEvent

This is an event that switches a proposition on or off. These always come in pairs and will naturally define TimeBlocks.

- *proposition* : **Proposition** //the proposition being switched
- *onOff* : {on , off} // on switches from FALSE to TRUE // off switches from TRUE to FALSE

B.2.3.10 ProbeEvent

This is a "measurement" type of event that determines whether a **Propostion** is true or false at a certain point in time. These **Event**s can occur due to sensors or because the coache herself presents information in **CoActual**. Actually performing the measurement may either generate the event or change the status of a planned **ProbeEvent**.

- *value* :**Boolean** // FALSE or TRUE

B.2.3.11 SingularEvent

There may be singular event that changes the state of the coachee but do not have a clear counterpart. An example is taking a dose of caffeine through coffee or chocolate, etc. We have included these to signal that they may need to be considered future. **SingularEvent**s may, in the future, probably be meaningfully related to **Score**s. Currently **SingularEvent**s may only enable us to specify simple **Constraint**s. Note that it may be possible to model a simple **Constraint** for a case like caffeine intake just as well with the help of switch events. (However, this may be unable to model the effect double doses). Currently, the class **SingularEvent** inherits from **Event** and has no additional selectors. An example of a singular Event might be;

- *caffeine* : **SingularEvent**

B.2.3.12 Circumstance

Circumstances one can be in. **Circumstance**s are **Proposition**s that can be true at the same time. E.g. one might be at home in bed watching TV in the bedroom, with the light on, which also is a Screen activity. The class **Circumstance** inherits from **Proposition** and has no additional selectors. We can, however, for the sleep domain, enumerate various circumstances as unique and identifiable inhabitants of the class Circumstance.

- *inBed* : **Circumstance**
- *atHome* : **Circumstance**
- *seeTV* : **Circumstance**
- *seeScreen* : **Circumstance**
- *inBedRoom* : **Circumstance**
- *lightHigh* : **Circumstance**
- *noisy* : **Circumstance**
- *inTravel* : **Circumstance**
- *atOffice* : **Circumstance**
- *awake* : **Circumstance**

B.2.3.13 Assigned

There are a number of **Assigned** activities/**Occupation**s that may be planned for the coachee by the coach. The class **Assigned** inherits from **Occupation** and has no additional selectors. We can, however, for the sleep domain, enumerate various shared occupations as unique and identifiable inhabitants of the class Assigned.

- **relaxationExc : Assigned**
- **sleepdiary : Assigned**

- **briefing : Assigned**
- **intake : Assigned**
- **cognitiveTherapy : Assigned**

B.2.3.14 Shared

There are a number of personal yet **Shared Occupation**s that the Insomnia-Coach may be informed about. The class **Shared** inherits from **Occupation** and has no additional selectors. We can, however, for the sleep domain, enumerate various shared occupations as unique and identifiable inhabitants of the class Shared.

- *eat* : **Shared**
- *play* : **Shared**
- *work* : **Shared**
- *sleep* : **Shared**

B.2.3.15 userSchedule

This is a time **Schedule** of planned **Assignment**s and **Shared** user **Occupation**s that can be seen and edited by the coachee. This also contains information on the **Occupation**s of the coachee that are **Shared**. We may be able to read in this **Schedule** when the coachee was watching TV, eating, awake, etc. So this time **Schedule** reflects facts and estimations of (changes in) the **Situation** of the coach at various points in time.

B.2.3.16 feelSchedule

This is a time **Schedule** that reflects the development of subjective **Feel** parameters in the past, and which may also contain projections that try to indicate how she might **Feel** in the (near) future.

B.2.3.17 thoughtSchedule

This is a time **Schedule** that reflects the development of subjective **Thought** parameters in the past, and which may also contain projections that try to indicate how she might think in the (near) future. This thought parameter reflects the progress or current status of the coachee's **AutomaticThoughts**. **Thought** parameter and the **thoughtSchedule** are new in the DepressionCare and they are used to identify the progress of the coachee in terms of thinking. The values of the **Feel** and **Thought** may come from the **ActivityDiary** and **ThoughtDiary Tool**s.

B.2.3.18 dealSchedule

This is a time **Schedule** that reflects when certain **Contract**s where agreed to, or when they were retracted, and how they were adhered to in the mean time. All these **Schedule**s use essentially the same time parameter and can therefore be overlaid on one another. So one can also combine all the **Schedule**s in one large **Schedule** that allows different views. The implementation of **Schedules** is quite straightforward. They consist basically of lists of entries referring to **Shared Occupation**s or **Assignment**s that relate to blocks of time or events that relate to points in time.

B.3 CoActual

The **CoActual** class contains the following information:

B.3.1 feel

List<**Feel**>, a collection of personal subjective "feeling" **Score**s like *stress, general wellbeing, subjective sleeping experience or fitness.* These parameters may be important for the coach to get some feedback on the effect of the therapy, and also may enable the coach to act empathically when the coachee is having a hard time. The **Feel** may be supported by a **Tool** that makes these parameters «Touchable».

- stress : Score //How stressed the coachee is
- unhappiness : Score //How unhappy the coachee is
- sadness : Score //How sad the coachee is
- guilt : Score //How guilty the coachee ise
- worthlessness : Score //if the coachee has the feeelings of worthlessness and if so how much
- tiredness : Score //How tired the coachee is
- weight change : Score //any change in the coachee'sweight
- sleepQuality : Score //Perceived sleep quality wellbeing : Score // General wellbeing
- fitness : Score // How fit do you feel?

B.3.2 thought

List<**Thought**>, a collection of personal subjective **Thought Score**s in different situations. This is new in the DepressioCare and used to see the progress in the coachee's thinking process. The coach has a list of possible automatic thoughts that a coachee might have in depression. He creates this list from his knowledge and previous coaching histories. He checks the frequency of this type of thoughts in the coachee's entries in the ThoughtDiary and on the basis of the frequency, he rates the coahee's thoughts. Some examples of automatic thoughts. The coach can also use automatic thoughts questionnaire to know more about the coachee's thinking.

- I feel like I am against the world.
- I am no good
- Why can't I ever succeed
- No one understands me
- I wish I were a better person

B.3.3 sit

List<**Situation**>, a collection of personal objective information concerning location, awake, eating, etc. The **Situation** represents aspects of the current **Situation** of the coachee that the coachee itself can easily and reliably judge or estimate, e.g. **Circumstance**s like *being awake, in bed or not, watching TV*, personal **Occupation**s like *eating* or *working*, as well as activities assigned by the InsomniaCoach like and timing estimates like at what time she hit the pillow or rose in the morning. The **Situation** may be supported by a **Tool** that allows the coachee to confirm, correct, or update the relevant values. Some of these values relevant to the sleep domain are indicated below:

- *inBed* : **Circumstance**
- *atHome* : **Circumstance**
- *seeTV* : **Circumstance**
- *seeScreen* : **Circumstance**
- *inBedRoom* : **Circumstance**
- *lightHigh* : **Circumstance**
- *noisy* : **Circumstance**

- *inTravel* : **Circumstance**
- *atOffice* : **Circumstance**
- *awake* : **Circumstance**
- *eat* : **Shared**
- *play* : **Shared**
- *work* : **Shared**
- *sleep* : **Shared** // might be signalled by hardware/actigraphy.
- *getPillowTime* : Daytime // estimation of time when you close your eyes to sleep (maybe unknown)
- *getRisingTime* : Daytime // estimation of time when you get up (maybe unknown)

B.3.4 tools

List<**Tool**>, a repertoire of tools that the coachee knows how to use.

B.3.5 identity

The **Identity** class contains *name*, *birth date, preference*s etc. this might also contain further info, like *profession.*

B.3.6 seen

List<**Name**>, the various concepts, **Technique**s, **Fact**s, **Interaction_Recipe**s the coachee is already familiar with.

B.3.7 deal

List<**Contract**>, a list of all actual agreements between the coachee and the coach

B.3.8 doubt

List<**Contract**>, a list of agreements proposed or promoted by the coach that the coachee has not yet agreed to.