AFFECTIVE MUSIC SYNTHESIS

Conveying Affectiveness in Leading-edge Living Adaptive Systems

CALLAS

Project IST-34800

Deliverable D134 WP1.3

Deliverable
version 1.0 – 31 December 2008
Document. ref.: callas.D134.UOR.WP1.3.V1.0

Programme Name: ..... IST
Project Number: 34800
Project Title:..... CALLAS
Partners:..... Coordinator: ENG (IT)
Contractors:
VTT Electronics, BBC, Studio Azzurro, XIM, Digital Video, Humanware, Nexture, University of Augsburg, ICCS/NTUA, University of Mons, University of Teesside, Helsinki University of Technology, Université Paris 8, Scuola Normale Superiore di Pisa, University of Reading, Fondazione Teatro Massimo, HITLaboratory New Zealand

Document Number: callas.D134.UOR.WP1.3.V1.0
Work-Package: WP1.3
Deliverable Type: ..... Document
Contractual Date of Delivery: ..... 31 December 2008
Actual Date of Delivery: ..... 31 December 2008
Title of Document: ..... Affective Music Synthesis
Author(s): ..... Atta Badii, Ali Khan, David Fuschi

Approval of this report .....

Summary of this report: ..... Description of CALLAS components of WP11

History: .....

Keyword List: .....

Availability ..... This report is: public

Table of Contents

Executive Summary

Computer music usually sounds mechanical; hence, if musicality and music expression of virtual actors could be enhanced according to the user's mood, the quality of experience would be amplified. We present a solution that is based on improvisation using cognitive models, Case-Based Reasoning (CBR) and Fuzzy values acting on close-to-affect-target musical notes as retrieved from CBR per context. It modifies music pieces according to the interpretation of the user's emotive state as computed by the emotive input acquisition componential of the CALLAS framework [24]. The CALLAS framework incorporates the Pleasure-Arousal-Dominance (PAD) Emotional Model which reflects the emotive state of the user and represents the criteria for the music affectivisation process. Using previous instances of affective adaptations from a case repository, a selection of music samples are adapted according to a user's mood. The PAD values are used by the CBR subsystem for selection and retrieval of a suitable previously stored case in the CBR repository. The user's mood is represented along three affective axes: "pleasure", "arousal", and "dominance". Using combinations of positive and negative states for these affective dynamics, the following octants are used as reference emotive states of the user, "Exuberant", "Bored", "Dependent", "Disdainful", "Relaxed", "Anxious", "Docile" and "Hostile". This allows for a level of interactivity that makes way for an interesting environment to experiment and learn about expression in music.

1. Introduction

Computer music usually sounds mechanical and it would increase the Quality of Experience if musicality and music expression of virtual actors could be enhanced for user mood resonance for enjoyment. Research work carried out notably by Arcos and Mantras [7] whose SaxEx system uses Case Based Reasoning and fuzzy rules to model human musician's affective performance expertise on a saxophone and to re-use it in delivering a targeted affective quality in the thus improvised music. The interactive possibilities of SaxEx developed in NOOS language allow the user to choose among a variety of alternative choices that can invoke a desired affect on the resulting musical expression. For example the users can express their preference along three affective dimensions: "tender-aggressive", "sad-joyful", and "calm-restless". This interactivity makes it a very interesting environment to experiment and learn about expression in music. Callas researchers acknowledge that following musical rules, no matter how sophisticated and complete they may be, is not enough to give added expression to otherwise mechanical sounding computer music. The challenges in deploying virtual affective musicians is to engineer an effective way of using that tacit expressive knowledge that human musicians gain and use in improvising to affectivise when playing a score; this is the sort of personal "touch" that gives the music a particular targeted quality and is about knowing when and where not to play a note or add an ornamental note.

2. Objectives

The aim is to produce an Affective Music Synthesis component for the CALLAS Shelf. This system may then be integrated into a CALLAS Showcase to enhance a user's experience with the demonstrations interface. The system is to use Case Based Reasoning (CBR) to adapt a limited selection of music sampled according to a user's mood using previous instances of affective adaptations from a case repository.

The aim of this module is to produce an Affective Music Synthesis (AMS) module for the CALLAS Shelf which can synthesise music according to an emotional input. This module will be based on improvisation using cognitive models, case based reasoning (CBR) and fuzzy logic acting on close-to-affect target musical notes as retrieved from CBR per context.

The resulting module will be used in the CALLAS Showcase components, which can be summarised as follows.

- Augmented Reality for Art, Entertainment and Digital Theatre - this component aims to augment the expressive range of possibilities for performers by capturing their movement and voices for analysis before digitally replicating the performers to show the different affective performances.

- Interactive Installation for Public Places – This showcase aims to use the emotional states of visitors to an interactive installation to create a richer type of interaction that allow the installations to better engage the visitors, provide a novel platform for sharing experiences and to better support visitor decision-making.

- Next Generation Interactive Television – This final showcase aims to adapt media content according to affective input from the user, resulting in a more affective output to the screen most probably through the use of an Embodied Conversational Agent (ECA).

The Callas roadmap for this work is clearly based on improvisation using cognitive models, Case-Based Reasoning and Fuzzy values acting on close-to-affect-target musical notes as retrieved from CBR per context.

- Analyse the Callas ShowCase Requirements and decide on a repertoire of Affective musical expressions to support it (in e.g. theatre, open air, immersive TV)

- Use example expressive music segments works to generate the Affective music repository to deploy Case Based Reasoning

- Develop the CALLAS Affective Music Improvisation Engine using heuristics acting on Fuzzy values of affect parameters of the similar-to-target-affect musical notes as retrieved from the CBR. This CALLAS Affect Improvisation Engine will have access to basic music knowledge rule sets and the basic score plus the retrieved similar notes from the CBR and then apply the appropriate transforms to the notes to lend expression using the CBR notes parameters and fuzzy values.

- Apply Saliency-triggered dynamic evaluation of resulting Musical Affect within a controlled environment with a representative user group in order to assess the Affect Improviser's saliency-triggered Affect-Effectiveness Feature System per Evaluation.

- Run the Callas Incubator's self-optimising GA Engine on the Affective Music Improviser's Emotional Speech synthesiser using the results of 4 above; to keep the improviser components designs optimised and adapted to new musical genres, cultures and affects - keep breeding better improvisers ready to Run

3. Relevant Literature Review

There has been significant interest in emotionally organised music collections in the past.

3.1 SaxEx and JIG

Previous work along these lines exists in the form of SaxEx [7] which uses CBR and fuzzy logic to generate expressive performances of melodies based on examples of human performances [7].

For musical improvisation, the Jazz Improvisation Generator (JIG) [16] and SICOM [17] have both made attempts at getting a machine to solve this task. SICOM produces a MIDI file with two tracks, a soprano and bass line, and produces results which its creators are pleased with. As mentioned in their paper [17], the main issues seem to be their small case library (of 3 cases) and the efficiency of their retrieval mechanism.

The interactive possibilities of SaxEx [7] (a case based reasoning system for generating expressive musical performances) were developed in NOOS language [8] which allow the user to choose among a variety of alternative choices that can invoke a desired affect on the resulting musical expression. Although, NOOS is a language which can be used in a CBR system, it should be noted that it is not specifically built for CBR. Moreover, upon raising a query with the NOOS team, it was discovered that an interpreter for NOOS is no longer available.

3.2 SICOM

There exists a system known as SICOM [17] which creates music using CBR and stores it as a MIDI track. The cases in the system represent notes in the track and the system uses CBR to decide which notes would best fit with the previously played notes.

3.3 The Affective Music Remixer

The Affective Music Remixer [21] arranges musical collections through the use of short clips (5 – 45 seconds) to gather emotional information. Emotional data is captured via the use of GSR (Galvic Skin Response), foot tapping (to show how engaged a user is) and subjective evaluation data (forms filled in by the user regarding their likes and/or dislikes).

3.4 Pleasure-Arousal-Dominance Emotional Model

The Pleasure-Arousal-Dominance (PAD) Emotional Model [22, 23] from the mid 1990s (1995/1996) categorises emotions in eight forms:

- Exuberant
- Bored
- Dependant
- Disdainful
- Relaxed
- Anxious
- Docile

- Hostile.

The samples library can be eight pieces to match number of extremes in PAD emotional model, plus an extra one for “neutral” emotion. Could also be multiples of this number so that an equal number of samples per emotion is available. Picard et al. [20] back this idea from PAD that there are eight emotional extremes and describe the difficulty behind machine recognition of human emotion.

3.5 Relevant Development Environments

Max/MSP [19] is a visual programming language for developing multimedia applications. It could be very beneficial for processing multimedia and affectivising music samples. However, it also requires distribution licences and thus involves cost overheads.

Pure Data (Pd) [18] could be used as an alternative to Max/MSP. It is free and relatively easy to use in comparison with Max. However, it should be noted that it lacks in areas where Max excels e.g. freedom in terms of choice of languages for extern development, etc.

Caspian [2] is a free CBR system from University of Wales Aberystwyth, which uses CASL, a cased based language, to represent items in its knowledge repository. It requires an index of an enumerated type with a fixed definition, which does not allow for the complexities involved in the Human emotional pattern (details in section 5).

4. Elements of Work

- Analyse the showcase requirements in order to select segments of music suitable to support them in their environment. A suitable range of affective expressions should also be chosen and, together with the musical segments, will be used to generate the Affective Music Repository which will be deployed to the case based reasoning module. The details of these musical segments will be stored as MPEG-SMR, a symbolic music representation which can store every detail of an audio file in a way as easily accessible as data in an XML file [ref].

- Develop the CALLAS Affective Music Improvisation Engine (AMIE) which will have access to basic music knowledge rule sets, the basic score of music retrieved from CBR and the similar notes retrieved from CBR. AMIE will apply appropriate transforms to the notes to generate the affective music using the CBR notes parameters and fuzzy values which are stored as part of an AMIE Improviser.

- AMIE should also provide a method of evaluating an Improviser's results in order for sensible information to be returned to the CBR repository. This system should be tested against a user group in a controlled environment.

- Optimise AMIE's Improvisers in terms of design and implementation in order to allow it to better adapt to new music genres, cultures and affects.

5. Case Based Reasoning

This section aims to outline the principles behind Case Based Reasoning and its usage in the context of the Affective Music Synthesis component of the CALLAS project.

5.1 Introduction to Case Based Reasoning (CBR)

Case Based Reasoning (CBR) is a method used in computing to allow software to solve problems by recalling how similar problem were dealt with in the past. The software would be given a set of basic cases and a set of rules by which it can alter these cases. When given a problem, the software would find the most relevant case (or maybe multiple cases) and modify it to better solve the problem. This process can be shown in the following diagram.

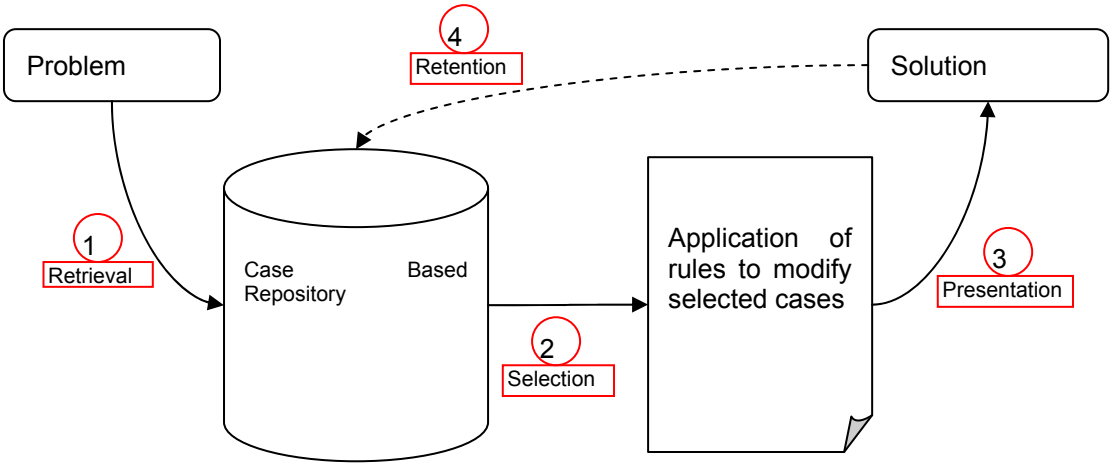


Figure 1: CBR Process

[1] The problem is entered into the system and analysed.

[2] The case which closest matches the details of the problem is selected.

[3] This case is modified to better fit the problem using predefined rules. This becomes the solution which is presented to the user.

[4] The solution is analysed for its effect and is stored for future use along with an indication of its successfulness. This allows the system to *learn*.

The system should be able to learn not only from its successful solutions, but also from its failed solutions. This is known as success driven and failure driven learning [1]. Successful solutions can be stored so that the solution can be reused without regenerating it. Solutions which failed to solve the problem can be used to generate better solutions for use in the future. The exact functionality of this concept depends on the implementation approach and details.

5.2 CBR in AMS CALLAS

The aim of the Affective Music Improvisation Engine (AMIE) is to play affective music based on a user's current emotion, which in turn should enhance the user's experience of CALLAS as a whole. By employing a case based reasoning approach and providing AMIE with a few basic cases and their solutions, the system should be able to adapt to a large part of the affective input domain.

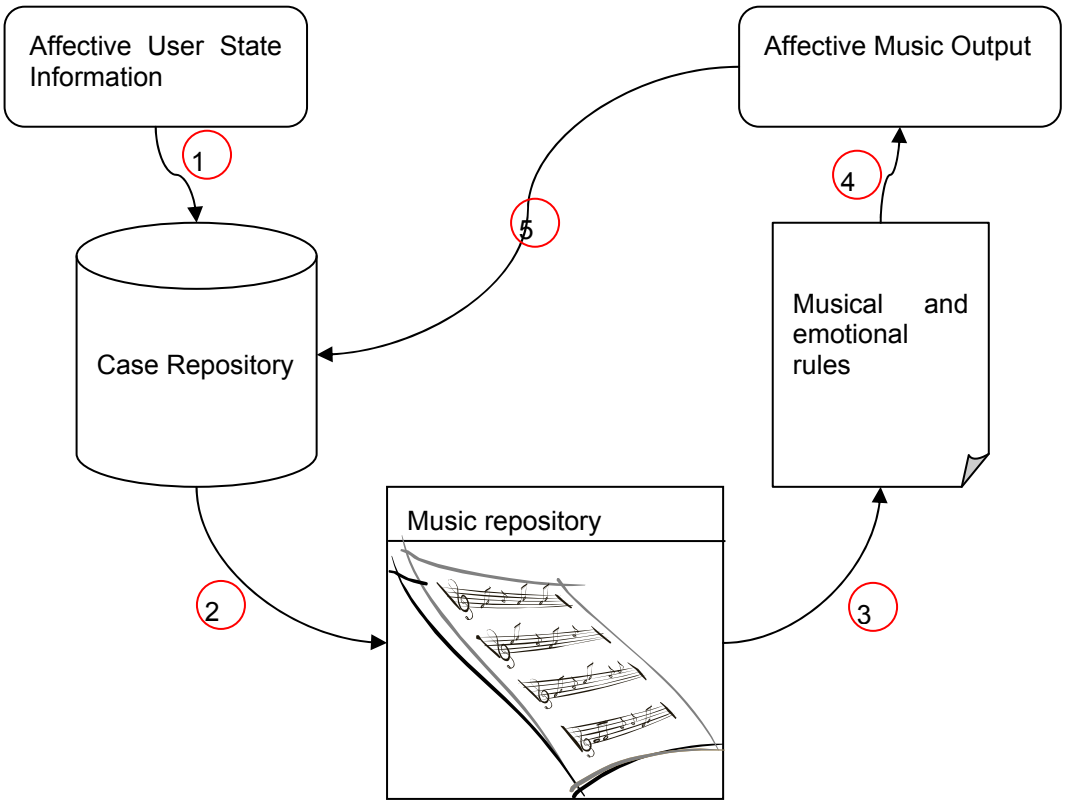


Figure 2: CBR Process in the Affective Music Synthesis CALLAS Component

[1] The input arrives from the multimodal CALLAS input module. As an example, this might contain information that the user is 70% happy and 30% calm.

[2] The best of the previous cases is chosen and modified accordingly to become the solution. The solution will contain information about which sample to select from the music repository and which rules to follow about how to modify this music.

[3] Music is selected from the repository according the solution's instructions.

[4] The selected music is modified using musical rules according to the solution's instructions. This forms the final affective musical output.

[5] The output is analysed for effectiveness and the solution is stored in the case repository.

It should be noted that in this example we have chosen to store the parameters used to

create the affective output (the base music and the rules by which it is to be altered) rather than storing the final musical solution (as an audio file). The parameters stored might be related to the speed to which the tune should be altered, the increase or decrease in song pitch, or other methods by which the song might be affectively altered. This method is considered by the authors to be an effective method in the case of AMIE as there will be many successful outputs possible for each case. By storing the method instead of the result, it is possible that different outputs of equal strength may be achieved, varying the music that a user might hear over time even if their mood remains constant. It may also be possible to better analyse the solutions; where the same method produces varying degrees of success it may be suggested that the method could be improved. Finally, where a particularly successful output has been produced, it might be sensible to store the output as an audio file which future cases will be able to modify directly. This would improve the output of the system over time, although it may also have an undesired effect and cause the same output to be repeated frequently.

5.3 Implementation decisions

Implementing a CBR system for CALLAS involves several tasks such as finding the closest solution to a given problem, defining the original cases, defining the rules by which to alter these cases, learning from successful cases, and more importantly, from unsuccessful cases etc.

Research into previous work found a small number of projects which have used Case Based Reasoning in a similar manner to the system planned here. The most relevant of these projects is SaxEx [7], which uses CBR to generate emotional musical performances. This also led to the discovery of NOOS [8], an object orientated language for writing Case Based Reasoning systems. Upon its discovery, it was planned to use NOOS to implement the CBR subsystem. However, the NOOS interpreter turned out to be unavailable and an alternative system, Caspian [9], was investigated.

A framework for defining cases and local repair rules exists as part of Caspian [2], a case based reasoning engine developed by the University of Wales Aberystwyth [2], which is intended to be reused in a variety of applications. It uses a case based language (CASL) for specifying cases, abstractions and repair rules. CASL is an extremely simple language for creating a case based system. It allows quick and easy creation of initial cases and repair rules for modifying those cases, as well as a method of abstracting case information for easier selection. CASL forces a user to index cases using an enumerated type of a fixed number of items. This indexing forces a user to choose a very specific value for a case property, rather than allowing for a non-discrete set of values by which a user might wish to index. This system also has the capability to rate each case for its relevance to a given problem.

By implementing the case based reasoning of AMIE using Caspian, there would be a need to create a case file in CASL (Caspian's case based language) which would represent the cases and rules required by AMIE. There would also be a need to turn the current Caspian executable into a DLL for easier integration with other modules of AMIE and CALLAS. It would be required to tie this together with the output system actually responsible for creating and playing the effective musical output.

Finally, a method for analysing the effectiveness of these solutions would need to be implemented. The system could be asked to store those solutions that are considered a success, and alter those that are considered failures. Once altered, these could be stored for use the next time a similar emotional context comes around. When any solution is reused, it could be re-analysed for success and modified or stored as necessary. This would lead to the solutions in the system gradually becoming more reliable until it becomes possible to choose a successful solution for any given emotional context.

Input into the Caspian module will be in the form of several numeric values representing the

perceived emotional context of the user, where output will be in the form of the music sample to play and the rules by which it is to be modified. Analysing the results may cause some difficulty, but it is expected that CALLAS emotive input modules will provide the user's new emotions. If the new emotions fall into an expected range given the choices Caspian previously made, then the solution might be considered a success. Otherwise, another music sample and/or set of rules could be chosen and stored for future use.

Having discussed what is involved in implementing a case based reasoning system for the AMIE system; Caspian initially looked to be an effective solution which would allow implementation of a CBR system for AMS in a minimal amount of time. However, further research showed that Caspian indexes its cases via a discrete set of pre-defined values which would be unsuited to represent the continuous range of possible emotional models. Further whilst the system might be alterable to index via the 8 octal ranges available in the PAD emotional model, this would add to a number of already time consuming tasks needed to alter Caspian for use with the Affective Music Improvisation Engine (AMIE). Therefore, it was decided to implement a custom CBR engine which would allow better integration with AMIE and would not pose any source and licensing issues,.

The CBR subsystem prototype has been written in C#. A flow diagram outlining the actions of the CBR system as well as class diagrams have been included in this document under the Implementation section.

The CBR subsystem is only one of three subsystems in AMS, with the Affective Music Improvisation Engine (AMIE) and the Analysis Engine being the remainder. A diagram explaining their roles within AMS can also be found later in this document, together with discussion on their design and implementation.

All three systems have been integrated successfully to demonstrate the way in which varying PAD model values i.e. the emotive input from the user affects the choice of cases and their generated solutions.

6. Pleasure-Arousal-Dominance Model

6.1 Abstract

Before implementation, research took place to learn about the emotional model which is expected to be used by the other CALLAS Shelf components. The chosen emotional model is known as the PAD model (Pleasure, Arousal and Dominance) and has been briefly documented in [11]. The implementation of this model was carried out in parallel with the CBR Subsystem and as a result is included as part of the CBR subsystem class diagram under the implementation section later in this document. This section aims to give an overview of how the PAD Emotional Model can be used within CALLAS with respect to AMIE (the Affective Music Improvisation Engine).

6.2 Emotional Models

An emotional model is used in order to effectively represent an emotional state. In the case of the PAD emotional model, 3 values representing Pleasure, Arousal and Dominance (PAD) are used. These three values allow for 8 extremes of state and many more intervening states [Table 1]. Whilst other emotional models may also be suitable for use with CALLAS, other CALLAS Shelf components have already selected this as their emotional model, making this the easiest form in which these components can transfer their data to AMIE.

Table 1: Octants of temperament space [6]

	P	A	D
Exuberant	+	+	+
Bored	-	-	-
Dependent	+	+	-
Disdainful	-	-	+
Relaxed	+	-	+
Anxious	-	+	-
Docile	+	-	-
Hostile	-	+	+

Definitions of above mentioned octants from Merriam-Webster Dictionary:

- *Exuberant* joyously unrestrained and enthusiastic
- *Bored* being weary and restless through lack of interest
- *Dependent* determined or conditioned by another; relying on another for support
- *Disdainful* full or expressing disdain (a feeling of contempt for someone or something regarded as unworthy or inferior)
- *Relaxed* freed from or lacking in precision or stringency; set or being at rest or at ease

- *Anxious* characterised by extreme uneasiness of mind or brooding fear about some contingency; ardently or earnestly wishing
- *Docile* easily taught; easily led or managed
- *Hostile* marked by malevolence; openly opposed or resisting; not hospitable; having an intimidating, antagonistic or offensive nature

According to [26], experiments have shown P, A, and D axes to give a beneficial description of temperament and personality. Eight basic octants (temperament types) are formed by combinations of these three dimensions of temperament as mentioned before:

- *Exuberant* (extroverted, outgoing, happy, sociable)
- *Bored* (sad, lonely, socially withdrawn, physically inactive)
- *Relaxed* (comfortable, secure, confident, resilient to stress)
- *Anxious* (worried, nervous, insecure, tense, unhappy, illness prone)
- *Dependent* (attached to people, needy of others and their help, interpersonally positive and sociable)
- *Disdainful* (contemptuous of others, loner, withdrawn and calculating, sometimes anti-social)
- *Docile* (pleasant, unemotional, and submissive; likeable; conforming)
- *Hostile* (angry, emotional in negative ways, possibly violent) [26]

6.3 Implementation

This emotional model will be implemented in AMIE as a class of its own with three properties representing P, A and D respectively. In order to allow the emotional models to be compared, a method will be required that calculates the difference between the PAD values. Pseudo-code for a simple implementation of this method can be seen in [Listing 1].

Since the PAD emotional model is in use by other CALLAS Shelf components, it is logical for AMIE to make use of it as well. Implementation of the model should be simple, requiring minimal effort in any object orientated language, whilst passing the model between components should require only the knowledge of the 3 separate values P, A and D.

Listing 1: Pseudo-code for calculating differences between PAD models

```
Func diff (pad1, pad2){
   diffP = pad1.P – pad2.P;
   diffA = pad1.A – pad2.A;
   diffD = pad1.D – pad2.D;
   return abs(diffP) + abs(diffA) + abs(diffD);
}
```

7. Design and Implementation of Affective Music Synthesis

7.1 Introduction

The task of the Affective Music Synthesis (AMS) system is to use Case Based Reasoning (CBR) to adapt a limited selection of music sampled according to a user's mood using previous instances of affective adaptations from a case repository. This section aims to outline the design and implementation of this system.

7.2 Design Overview

The AMS system contains 3 sub-systems; the Case Based Reasoning (CBR) system, the Affective Music Improvisation Engine (AMIE) and the Analysis Engine. The diagram below shows the input, output and interactions between the systems.

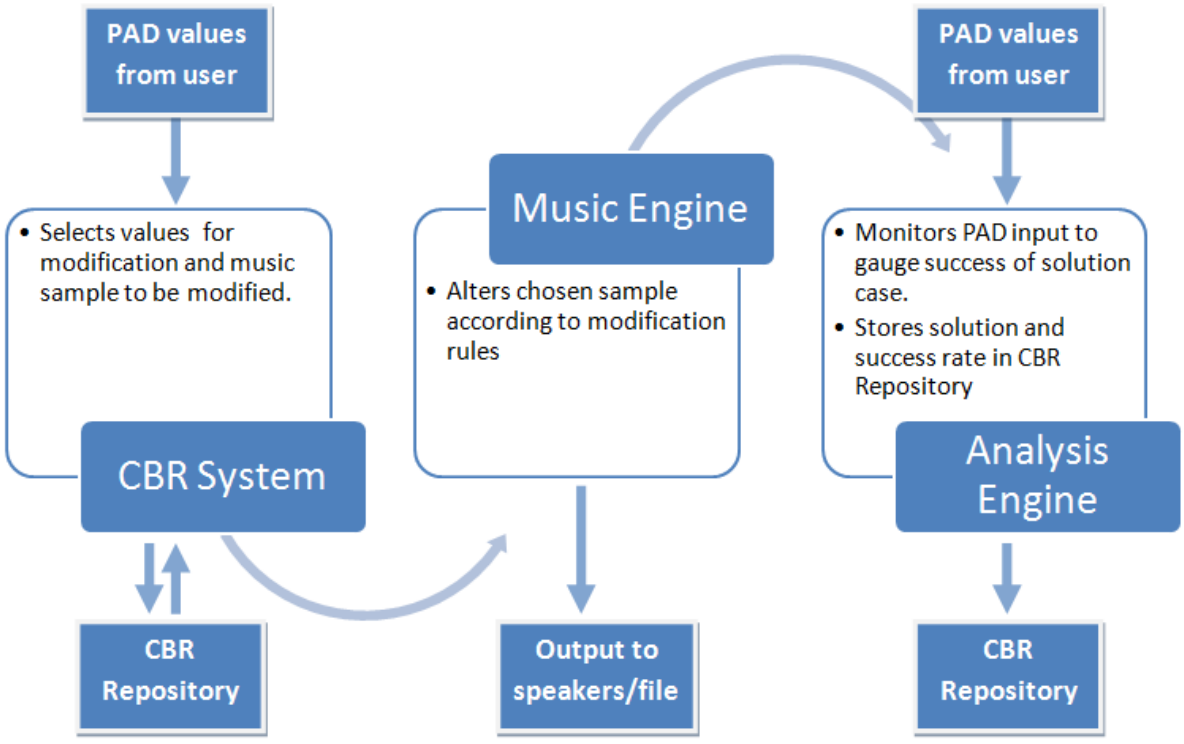


Figure 3: Input, Output and Interactions between the AMS sub-systems

As shown, the first point of entry for the PAD Values (acquired by the multimodal input modules of CALLAS) into the Affective Music Synthesis module is the CBR subsystem. These PAD values reflect the emotive state of the user and represent the criteria for the music affectivisation process. The PAD values are used by the CBR subsystem for selection and retrieval of a suitable previously stored case in the CBR repository (a music sample and the parametric values of its associated solution which are to be used for modification of the sample under consideration). The solution of the retrieved case is passed to the Affective Music Improvisation Engine (or AMIE) which uses the values from the solution to affectivise the given music sample. This generated music sample can be output either directly to the

speakers, or to a new file. To help improve the CBR system's selection of cases, the Analysis Engine uses further input PAD values to modify the selected case and store the solution in the repository.

The system is implemented in C# and currently uses only a single external library known as MIDI for .NET (Toub.Sound.Midi) [4]. As indicated, this shows that the system currently uses MIDI files for its musical input and output.

7.3 The CBR System

The CBR system contains 4 classes, whose properties and methods are shown in the diagram below.

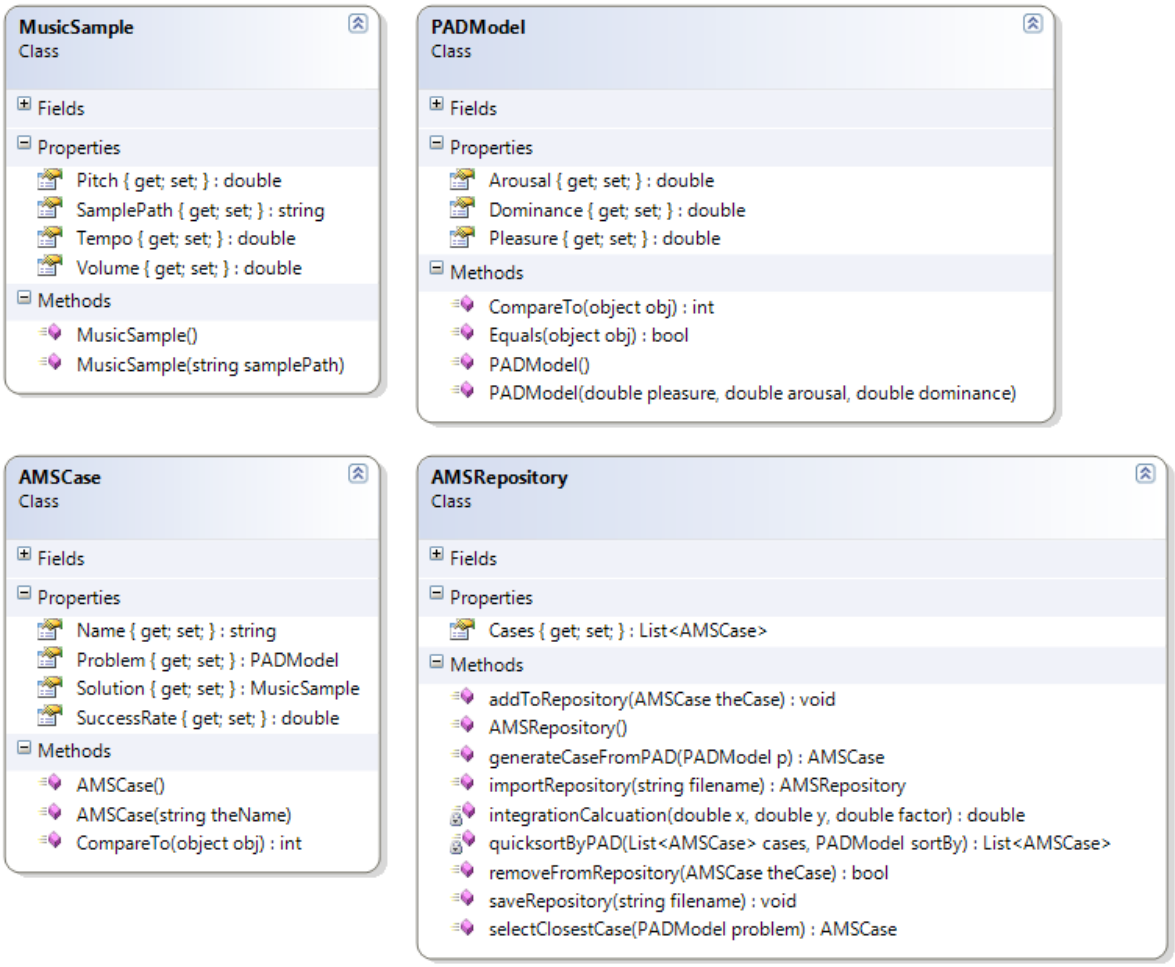


Figure 4: CBR System Classes

AMSREPOSITORY This class stores a collection of **AMSCASES** and offers access to these cases via a variety of methods. The public methods are described as follows.

addToRepository(...)	Adds an **AMSCASE** to the repository.
removeFromRepository(...)	Removes the first matching **AMSCASE** from the repository. Returns a value

	indicating success.
selectClosestCase(...)	Takes in a **PADMODEL** and returns the closest matching solution in the repository.
generateCaseFromPAD(...)	Takes in a **PADMODEL** and returns either an existing case that matches this model, or a newly generated case with suitable parameters.
saveRepositor**SITORY**(...)	Saves the repository to a file.
importRepository(...)	A static method used to import a repository from a file. Returns a new **AMSREPOY**.

AMSCASE	This class represents a case with a problem definition (the **PADMODEL**) and a solution (the **MUSICSAMPLE**). **AMSCASES** can also be given a name and a success rate.
PADMODEL	This class stores the three values of a PAD (Pleasure, Arousal, Dominance) emotional model and provides methods for checking the equality of instances.
MUSICSAMPLE	This class stores the location of a music sample, and three values (Tempo, Pitch and Volume) which can be used as information about the music sample or the alterations to make.

Significant algorithms in this system include the **PADMODEL** comparison algorithm. This algorithm maps each **PADMODEL** onto a three dimensional space and calculates the "distance" between these points using standard geometrical equations. These equations involve the use of the square root function, so a sign indicating whether one **PADMODEL** is less than or greater than another must then be calculated. This is done by subtracting the axis coordinate values from each other and determining the overall sign adding these differences together and determining the sign of the result.

The second significant algorithm is the case generation algorithm found in the **AMSREPOSITORY** class. This algorithm uses a quick sort algorithm to order the cases according to their "closeness" to a given **PADMODEL**. The "closest" case to the problem can then be selected and, should it match the problem precisely, be returned. Should this case not precisely match, then the generation becomes more complex. In this circumstance, a case on the "opposite side" of the problem model is selected. For example, if our closest solution has a problem model that is considered "less than" the actual problem, then the second selected case should be considered "greater than". With two suitable cases selected, the solutions in these cases can be integrated with respect to the proportion of their respective distances from the problem case. For instance, when the closest case is 2 units away from the problem and the second unit is minus 4 units from the problem, the entire first case is added to one third (the distance of the closest case from the problem / the total distance between the cases) of the second case. When a second case cannot be found for integration the solution returned is the closest case.

Improvements to the **AMSREPOSITORY** class could be to improve the selection and generation of cases from a **PADMODEL**. This could also require alterations to the **PADMODEL** class to improve the comparison algorithm. The major improvement required for case generation is an algorithm to generate a case when there are no cases on the "opposite side" of the problem domain, but other performance enhancements could also be sought.

7.4 The Affective Music Improvisation Engine (AMIE)

AMIE is a simple system with only a single class which accesses the Toub.Sound.Midi library. The methods contained within this class are discussed below. All methods in this class are accessible without creating an instance.

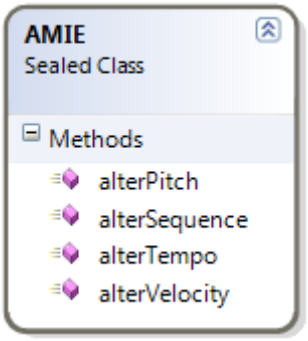


Figure 5: AMIE Class

alterSequence(...)	This requires a **MUSICSAMPLE** and three values to be passed in which reflect how to alter the pitch, tempo and velocity (volume) of the music sample. Once altered, the music sample is returned to the caller.
alterPitch(...)	Alters only the pitch of a passed in **MUSICSAMPLE**.
alterTempo(...)	Alters only the tempo of a passed in **MUSICSAMPLE**.
alterVelocity(...)	Alters only the note velocity (volume) of a passed in **MUSICSAMPLE**.

As a result of its simplicity and the use of the Toub.Sound.Midi library, there are no particularly complex algorithms in AMIE. The pitch alteration algorithm simply uses a method from the external library to alter the pitch. The tempo alteration algorithm merely requires the discovery and alteration of a single attribute of the given sample (although improvements to this algorithm can be made by looking for other places this attribute occurs). Finally, the velocity alteration algorithm simply loops through each note in the music sample setting a new note velocity. An improvement to this final algorithm would be to recognise that each note may be given a different volume and alter the velocities accordingly.

A major improvement to AMIE could involve the ability to affectivise musical pieces in an improved manner by employing MPEG-SMR [20] for a comprehensive affectivisation parameter set. Other improvements could involve addition of new effects such as using different “backing tracks” or “flourishes” into the sample to alter the emotional context of the music.

7.5 The Analysis Engine

The Analysis Engine consists of a single class with a single static method. This method analyses the success of a solution by using the passed in **PADMODEL** to alter the solution case. The altered case is then returned to the caller for inclusion into the CBR repository.

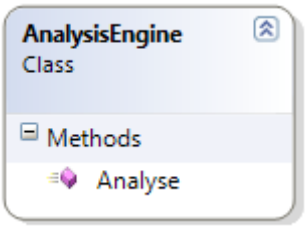


Figure 6: Analysis Engine Class

The analysing algorithm modifies the original case so that its **PADMODEL** is equal to the feedback. Improvements to be made here will include accounting for other times this case has been supplied, as well as possible modifications for the cases used to generate this case.

7.6 Integration

To integrate the AMS modules with a new program, references must be added to four DLLs; CALLAS_AMIE.dll, CALLAS_CBR.dll, AnalysisEngine.dll and Toub.Sound.Midi.dll. It is intended as part of future refinements that the number of DLLs required will be fewer, however as the source code is expected to undergo more changes, this separation is required.

Once the DLLs have been referenced, classes can be instantiated and public methods and properties listed above can be used. For instance, creating a new **AMSREPOSITORY** from a file can be achieved in C# using the following code.

Listing 2: Pseudo-code for creating new AMSRepository from a file

```
AMSRepository repository = AMSRepository.importRepository(filename);
```

Alternatively, a new **AMSREPOSITORY** could be created using the following code.

Listing 3: Pseudo-code for creating new AMSRepository

```
AMSRepository repository = new AMSRepository();


AMSCase case1 = new AMSCase("Neutral");
case1.Problem = new PADModel(0, 0, 0);
case1.Solution = new MusicSample("saints.midi");
repository.addToRepository(case1);
```

As one last example, the following code can be used to retrieve a new case from the repository which represents a **PADMODEL** with 0.1, 0.5 and -0.2 as its values for Pleasure, Arousal and Dominance respectively.

Listing 4: Pseudo-code for retrieving a new case from AMSRepository

```
PADModel padmodel = new PADModel();
padmodel.Pleasure = 0.1;
padmodel.Arousal = 0.5;
padmodel.Dominance = -0.2;
AMSCase generatedCase = repository.generateCaseFromPAD(padmodel);
```

Further code examples can be found in the source code for the AMS Demonstration Program.

7.7 CBR System Flow Diagram

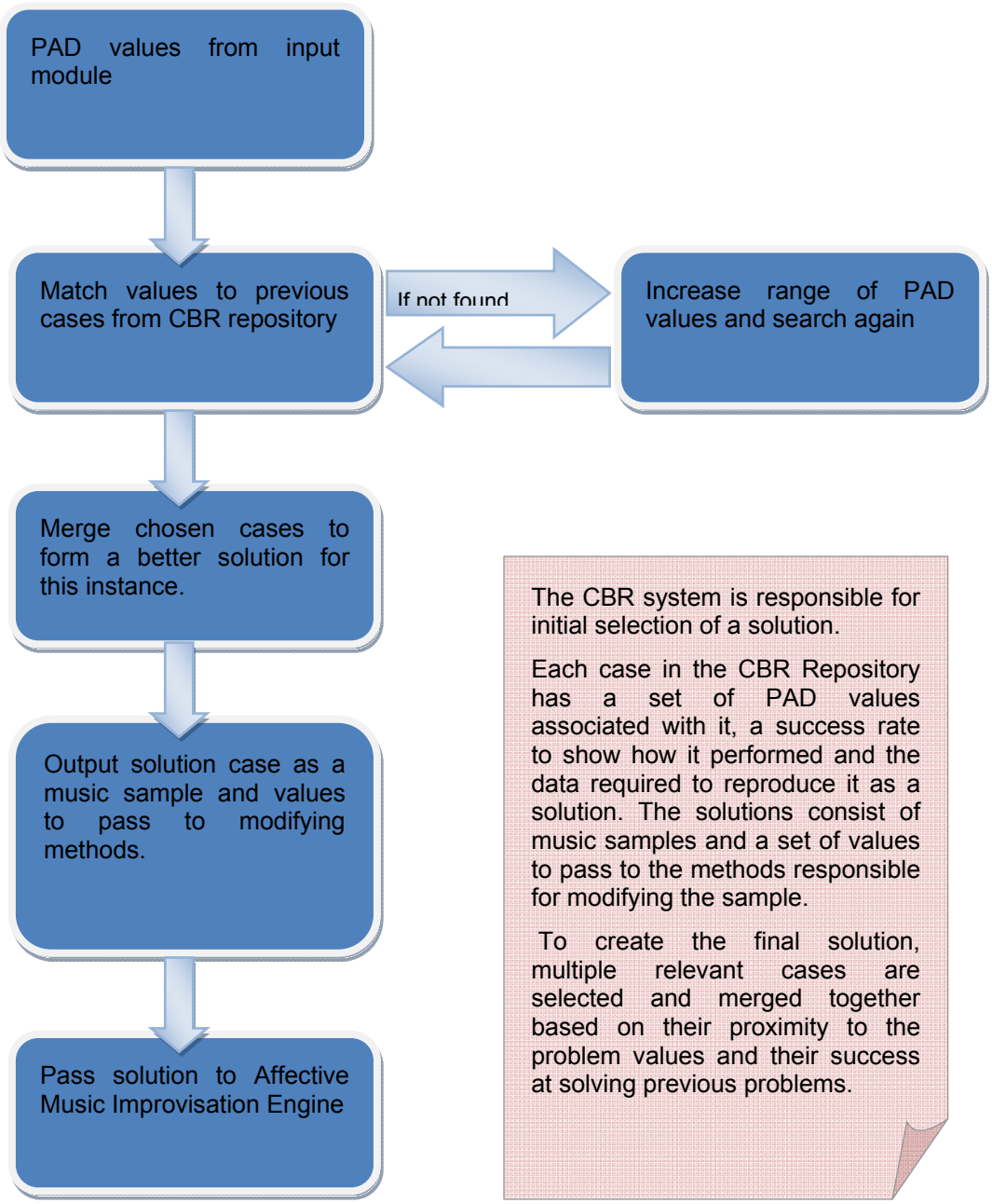


Figure 7: CBR Flow Diagram

8. Standalone AMS Demonstrator

8.1 Introduction

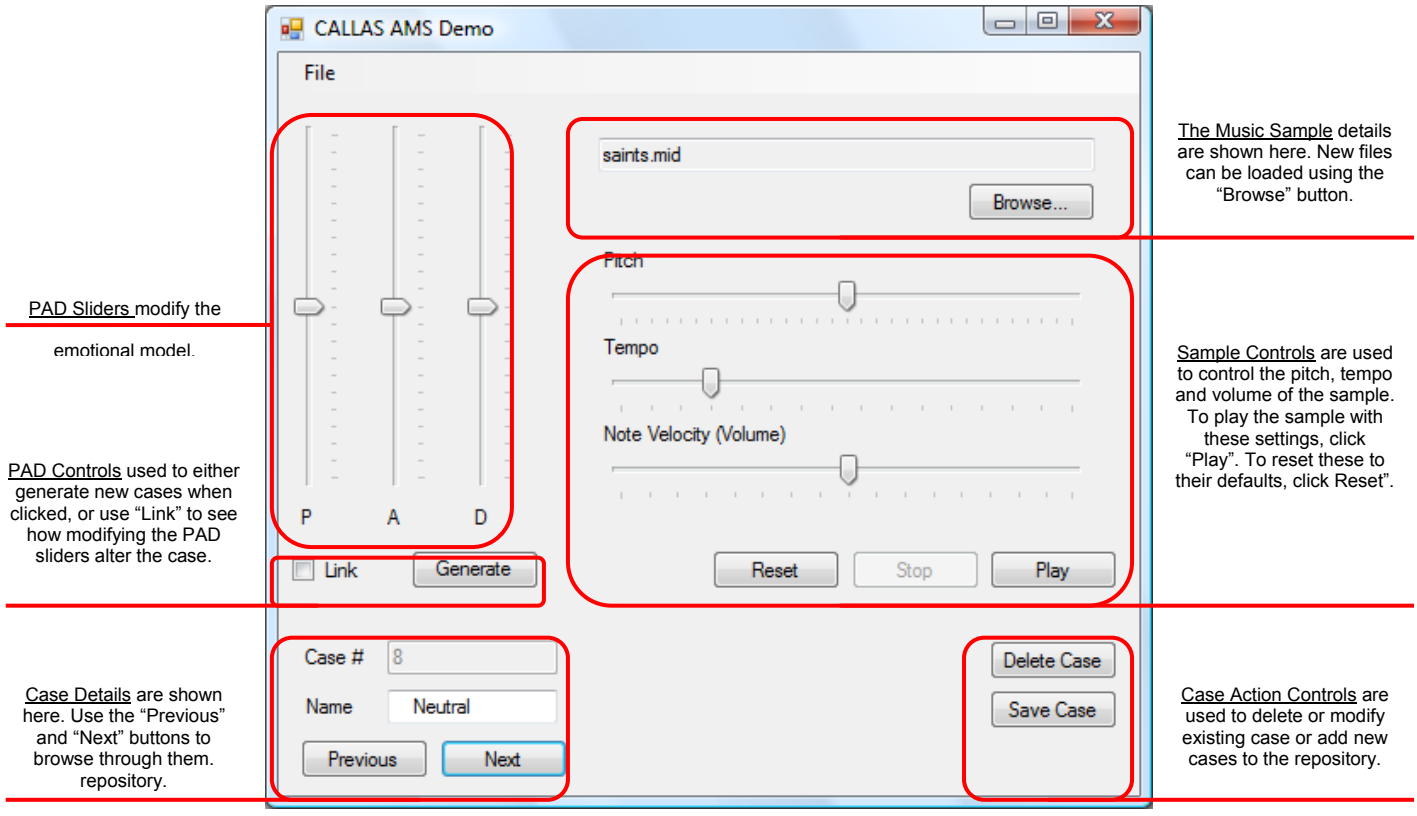
The purpose of this standalone program is to demonstrate the capabilities of the Affective Music Synthesis module of CALLAS and its 2 sub-systems; Case Based Reasoning (CBR) and the Affective Music Improvisation Engine (AMIE).

The current standalone program requires no installation as long as its files are kept together in an uncompressed folder. Furthermore, this prototype working in a single machine environment has no reported errors and bugs.

8.2 Interface overview

This section gives a brief overview of the main sections of the program interface.

The Pad Sliders indicate the levels of each value in the PAD (Pleasure, Arousal, Dominance) emotional model. By enabling the “Link” checkbox, the user can modify the PAD values and verify the behaviour of the automatic case generation. However, in order to modify the PAD values for a specific case, it is recommended that the “Link” checkbox is disabled. The PAD value sliders and sample controls can then be modified separately, and clicking “Save Case” will save the case to the repository. Cases can also be deleted from the repository by using the “Delete Case” button with a selected case. When the “Case #” textbox is empty, the case does not currently exist in the repository. The user has the option to give this new case a name before clicking “Save Case” to enter the case into the repository. It is recommended that all user generated cases should be given names and if a user attempts to enter a case without a name into the repository, they will be asked to verify their action.



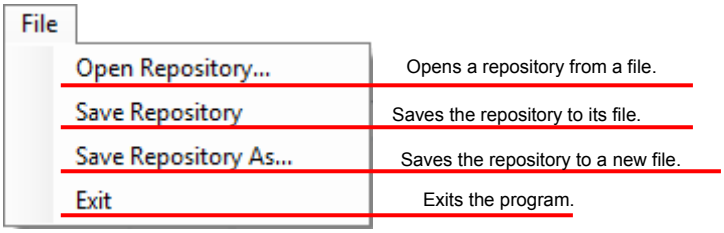


Figure 8: Standalone AMS Demonstrator GUI

To save the repository or load a new repository from a file, the "File" menu is used. By default, the program will open with a preset repository stored in the program folder. Unless the user has opened a new repository or saves the repository using "Save Repository As...", this is where the repository will be saved. Repository files are XML files whose specification will be discussed in the next section.

8.3 XML Specification

An example of a saved repository file can be seen below. The file contains a number of **AMSCase** elements within a single **AMSRepository** element. Each **AMSCase** element then has a **PADModel** element representing the problem and a **MusicSample** element representing the solution. These elements then have attributes which describe their state. Friendly names for each **AMSCase** can also be stored as an attribute of the **AMSCase**.

```
<?xml version="1.0"?>
<AMSRepository>
  <AMSCase Name="Exuberant">
    <PADModel Pleasure="1" Arousal="1" Dominance="1" />
    <MusicSample SamplePath="saints.mid" Pitch="7" Tempo="0.5" Volume="100" />
  </AMSCase>
  <AMSCase Name="Bored">
    <PADModel Pleasure="-1" Arousal="-1" Dominance="-1" />
    <MusicSample SamplePath="saints.mid" Pitch="-3" Tempo="2" Volume="50" />
  </AMSCase>
```

PADModel elements have 3 attributes (**Pleasure**, **Arousal** and **Dominance**) all of which must contain a numerical value between -1 and 1. These may if necessary be decimal values. **MusicSample** elements contain 4 attributes. The **SamplePath** attribute contains a path to a music file to be played which can be either a relative path, or a full path. The **Pitch**, **Tempo** and **Volume** attributes contain the modifications which are to be made to the music sample. These may be between any values, however for the purposes of the AMS Demonstration program these values will need to be between the following values.

Pitch	-15 to 15 (where 0 is no change to the Pitch)
Tempo	0 to 4 (where 1 is no change to the tempo)
Volume	0 to 100 (where 50 is no change to the volume)

9. TCP Integration: AMS Server and Demo TCP Client

9.1 Introduction

In order to facilitate integration with other CALLAS Shelf components, a TCP/IP based interface has been developed by the Affective Music Synthesis component. Any client/component may interact with the system through a set of simple commands issued via a TCP connection. This section outlines the working of this method.

```
CALLAS - Affective Music Synthesis

 Engine starting...

Please specify port (to use default port: 3000, press Enter)

Please specify whether you require RMA service: [ON = 1 (default)]; [OFF = 0]
0
Starting server using port 3000
RMA service: 0
Waiting for a connection...
```

```
CALLAS - Affective Music Synthesis

 Engine starting...

Please specify port (to use default port: 3000, press Enter)

Please specify whether you require RMA service: [ON = 1 (default)]; [OFF = 0]

Starting server using port 3000
RMA service: 1
Waiting for RMA connection...
```

9.2 TCP Connections

To start the AMS TCP server, it is possible to either run the executable directly, or start the application through the command line in order to specify a port for TCP communication. For instance, to start the application using port 1616, use the command “ams_tcp 1616”. The default port for the application, when none is specified, is port 3000.

To connect to the server, simply open a TCP socket to the selected port. For instance, if the TCP server is running on the local host using port 3000, the C# code for connecting would be as follows.

Listing 5: TCP Connections

```
using System.Net.Sockets;
using System.IO;

public class AMSClient
{
        public void connect()
        {
                TcpClient client = new TcpClient();
                client.Connect("127.0.0.1", 3000);

                //We can now use a StreamReader to read from the server
                StreamReader sr = new StreamReader(client.GetStream());
                String message = sr.ReadLine();

                //We can also use a StreamWriter to write to the server
                StreamWriter sw = new StreamWriter(client.GetStream());
                sw.AutoFlush = true;
                sw.WriteLine("exit");
        }
}
```

There is no need to send a command to the server to connect. However, when the application has finished using the client, the "exit" command should be sent. This will allow the server to free up resources used by the connection and be better able to deal with new incoming connections.

All commands sent to the server should be terminated with a new-line character. The use of a new-line character within a command or its parameters will cause unexpected behaviour.

It should also be noted that the server assigns different case repositories to each connection; changes made by one client will not affect any others. Loading and saving repositories will be covered in the next section.

9.2.1 Using AMS Server's Real-time Affectivisation Service

Once connection has been established, the AMS Server requests the user to specify whether the service required is for real-time affectivisation or batch mode. By default, the server extends real-time affectivisation services to its connected client. If real-time generation of affectivisation parametric values is required by the client for music affectivisation in real-time, the AMS Server opens a second port for outgoing messages. Hence, two half-duplex communication ports are opened between the client and the server. More details about the real-time AMS client (RMA patch developed in Pure data) can be found in section 10.

9.3 Manipulating Repositories

Once connected, the client has access to a copy of the server's default repository, specified by the file "repository.xml" in the server's runtime directory. To load a new repository, simply send the server the command "loadrepos" followed by an XML representation of the repository which will be discussed below. To save the repository, send the command "saverepos". The server will then transmit the XML representation of the repository to the client and the client can write this to the associated file.

9.4 Manipulating Cases

With the correct repository loaded onto the server, individual cases can now be accessed through the use of the "getcase", "savecase" and "delcase" commands. The "getcase" command needs only a number representing the index of the case to be retrieved from the repository. If this number is not valid, then the message "Invalid Index" will be returned. If the index is valid then an XML representation of the case will be returned in the form given in the XML example later in this document.

To save a case, the "savecase" command requires that an XML representation of the case be passed to the server as a parameter. If the XML includes an "Index" attribute then the case will overwrite the case found at that index. If there is no "Index" attribute or this attribute is less than zero, then the case will be stored as a new case in the repository and assigned an index number. When the case has been stored, the XML representation of the case will be returned, complete with an "Index" attribute to show its place in the repository.

Finally, the "delcase" command requires an index of the case to delete. For example, "delcase 5" would delete the 5th case in the repository.

9.5 Generating and Playing Cases

The main functionality of the Affective Music Synthesis system is to provide a method by which music can be generated based on an emotional model. To do this via the TCP server, the "generate" command should be used to create a new case and the "play" command used to play a case. "Generate" requires an XML representation of a new case with appropriate PAD values. These PAD values will be used to populate the other fields of the case and the resulting XML representation is returned to the client. "play" can then be passed this XML in order to play the resulting sound file.

9.6 Generating Cases for Real-time Affectivisation

To request the AMS Server to generate corresponding affectivisation parametric values from PAD model cases, the "realtime p.p a.a d.d" command should be sent; p.p, a.a. and d.d represent pleasure, arousal and dominance values respectively. When received by the server, XML representation of a sample case is created and processed by the server in conjunction with the CBR for matching retrievals and/or computation of parametric values. These values are then sent back to the real-time client in the form of "ptv p.p t.t v.v"; where p.p, t.t and v.v represent pitch, tempo and velocity i.e. affectivisation parametric values.

9.7 Protocol for communication with the AMS TCP Server

Table 2: Protocol for communication with the AMS TCP Server

Command	Description	Returns
getcase <int>	Returns the case at the specified index in the repository.	Returns the <xml> representing the case found at that position.
Savecase <xml>	Saves the case to the repository. If the <xml> contains a valid index, then the case will replace any found at that index in the repository.	Returns the <xml> including the index of the case in the repository.
delcase <int>	Deletes the case found at the	

	specified index in the repository.	
Generate <xml>	Generates a case based on the <xml> of an empty case with relevant PAD values.	Returns <xml> representing a case for these PAD values.
realtime <float> <float> <float>	Generates a case based on the PAD model received in the order, pleasure, arousal and dominance	Returns “ptv p.p t.t v.v” where p.p, t.t. and v.v represent floating point affectivisation parametric values representing a case for the input PAD values.
play <xml>	Plays the music associated with the <xml> case using the specified parameters in the <xml>	
loadrepos <xml>	Loads the <xml> representation of the repository into the server.	
Saverepos	Retrieve the <xml> representation of the repository.	Returns the <xml> of the server repository for use by the client, i.e. storage in a file.
Exit	Closes the connection with the server.	

9.8 Example XML

```
<?xml version="1.0"?>
<AMSRepository>
  <AMSCase Name="Exuberant" Index="0">
    <PADModel Pleasure="1" Arousal="1" Dominance="1" />
    <MusicSample SamplePath="saints.mid" Pitch="7" Tempo="0.5" Volume="100" />
  </AMSCase>
  <AMSCase Name="Bored" Index="1">
    <PADModel Pleasure="-1" Arousal="-1" Dominance="-1" />
    <MusicSample SamplePath="saints.mid" Pitch="-3" Tempo="2" Volume="50" />
  </AMSCase>
```

For the purposes of this TCP server, certain parts of the example XML are used for communication. For instance, the “loadrepos” and “saverepos” commands both use the full AMS XML specification; however the Index attribute of an AMSCase is optional in these circumstances.

The “play”, “generate” and “savecase” commands all require only a single AMSCase element to be sent. The Index attribute is again optional, unless the intention is to save a modified case in its original position in the repository.

```
<AMSCase Name="Bored" Index="1">
 <PADModel Pleasure="-1" Arousal="-1" Dominance="-1" />
 <MusicSample SamplePath="saints.mid" Pitch="-3" Tempo="2" Volume="50" />
```

All XML sent to the server should be done so without line breaks. Equally, all XML received from the server will be done without line breaks. It is considered that a line break is the signal for the end of a command.

9.9 AMS TCP prototypes

The current networked prototypes working in

i) A single machine environment (loopback) as well as

ii) Client/Server residing on separate machines

have no reported errors and bugs.

9.9.1 Single Machine Client/Server Environment using loopback

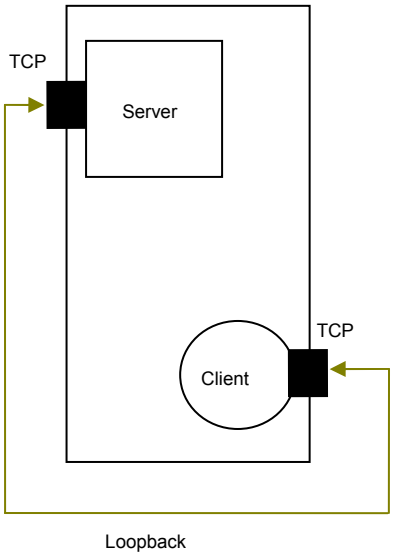
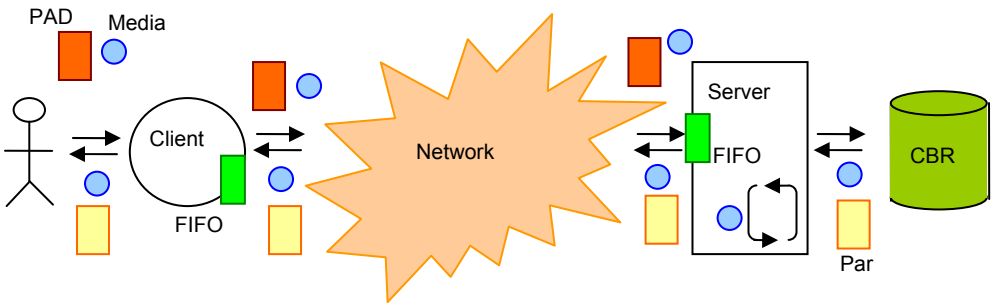


Figure 9: Single Machine Client/Server Environment using loopback

As seen in figure above, the TCP single machine version of AMS uses loopback to send PAD values from the client to the server. The server engine then either retrieves a matching case from the CBR or in case there is absence of a matching case, computes the necessary parametrics of affectivisation and sends them back to the client on the same machine via TCP loopback.

9.9.2 Multiple Clients in a networked environment connecting to AMS TCP Server



Server – Affective Music Synthesis TCP Server Engine
Client – Affective Music Synthesis TCP Client
CBR – Case Based Repository
PAD – Values for Pleasure-Arousal-Dominance Emotion Model
Media – Sound file to be affectivised
Par – Parameters of affectivisation i.e. tempo, pitch, velocity …
FIFO – Fist-In-First-Out list for job prioritisation

Figure 10: Multiple Clients in a network environment connecting to AMS TCP Server

The figure above demonstrates the working of TCP version of AMS when one or more clients connect to the server remotely for affectivisation of music pieces. As shown, a client may initiate connection to the listening server via TCP and send PAD values along with the media (sound file) to the server over a network, in order to use the Affective Music Synthesis Engine capabilities on the server side. The server retrieves a matching case (if one exists) from the CBR and provides the modified parameters of affectivisation to the client along with the affectivised media. After the job is complete, client initiates a request for the connection to be terminated via the *exit* command.

The temporal diagram below shows that in case of long-lasting interactions it could happen that at time of reply the communication may have been lost or other problem may have occurred, thus it would be better to pass from a synchronous approach to an asynchronous one (see red acks).

In the fully networked and distributed scenario it may be better to have I/O queues (in green) to handle the process especially in case of concurrent events and also to grant scalability.

It may be profitable to pass from a situation in which the command line transferred includes parameter to a situation in which files are transferred (as jobs) holding the parameter and, if needed, also the actual file via FTP; in this manner it would also be possible to further exploit the flexibility introduced by the queues as we could adopt a job oriented approach that could be easily parallelized.

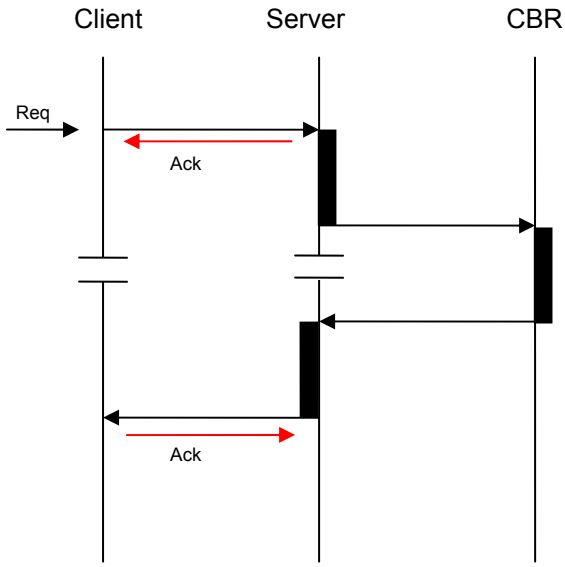


Figure 11: Client-Server-CBR Interaction Diagram

9.10 Screenshots of AMS TCP Server

Screenshots of above mentioned commands sent to the AMS server by a demo TCP client and their respective responses.

GENERATE <xml>

```
Message Recieved: generate <AMSCase Name="" Index="-1"><PADModel Pleasure="-0.2"
 Arousal="-0.47" Dominance="0.1" /><MusicSample SamplePath="" Pitch="0" Tempo="0
" Volume="100" /></AMSCase>
Sending <AMSCase Name="" Index="-1"><PADModel Pleasure="-0.2" Arousal="-0.47" Do
minance="0.1" /><MusicSample SamplePath="saints.mid" Pitch="-2.10349708818434" T
empo="1.30049958402633" Volume="79.0249792013167" /></AMSCase> to the client.
```

Figure 12: Generate <xml>

GETCASE <int>

```
Message Recieved: getcase 0
Sending <AMSCase Name="Neutral" Index="0"><PADModel Pleasure="0" Arousal="0" Dom
inance="0" /><MusicSample SamplePath="saints.mid" Pitch="0" Tempo="1" Volume="64
" /></AMSCase> to the client.
Message Recieved: getcase 1
Sending <AMSCase Name="Bored" Index="1"><PADModel Pleasure="-1" Arousal="-1" Dom
inance="-1" /><MusicSample SamplePath="saints.mid" Pitch="-3" Tempo="2" Volume="
50" /></AMSCase> to the client.
Message Recieved: getcase 2
Sending <AMSCase Name="Relaxed" Index="2"><PADModel Pleasure="1" Arousal="-1" Do
minance="1" /><MusicSample SamplePath="saints.mid" Pitch="0" Tempo="1.5" Volume=
"34" /></AMSCase> to the client.
_
```

Figure 13: GETCASE <xml>

SAVECASE <xml>

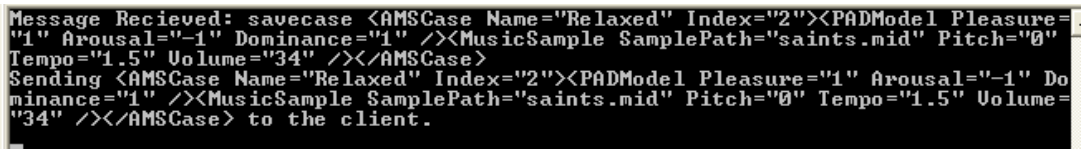


Figure 14: SAVECASE <xml>

PLAY

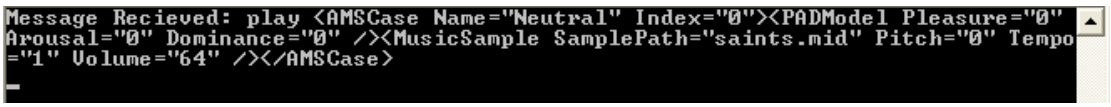


Figure 15: PLAY

SAVEREPOS

Figure 16: SAVEREPOS

LOADREPOS <xml>

Figure 17: LOADREPOS <xml>

EXIT

Figure 18: EXIT

Screenshots of AMS TCP Server version 2 and a demo TCP Client (also part of AMS TCP prototype version 2) connecting to the server for affectivisation services:

```
CALLAS - Affective Music Synthesis

 Engine starting...

Please specify port (to use default port: 3000, press Enter)

Please specify whether you require RMA service: [ON = 1 (default)]; [OFF = 0]
0
Starting server using port 3000
RMA service: 0
Waiting for a connection...
```

Figure 19: CALLAS Affective Music Synthesis TCP Server

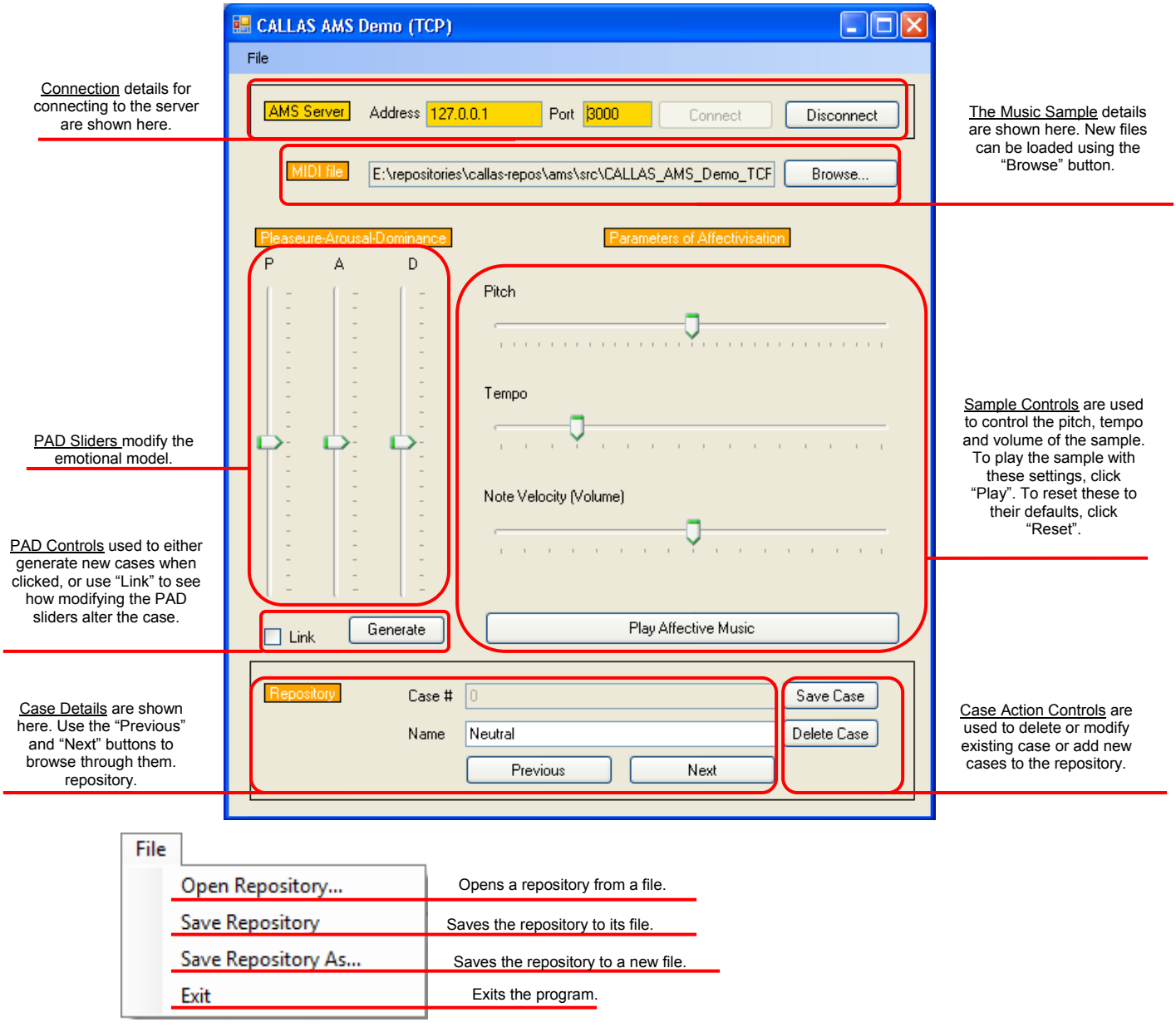
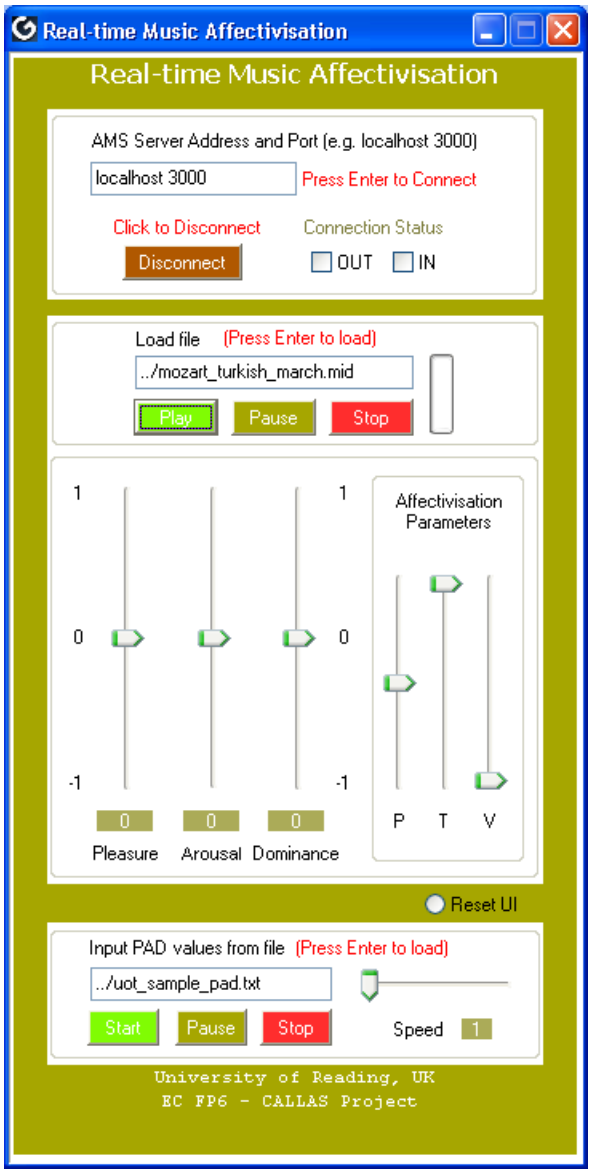


Figure 20: CALLAS AMS TCP Server – Demo Client GUI

10. Real-time Music Affectivisation (RMA)

10.1 Introduction

To carry out affectivisation of music on-the-fly, a number of audio programming languages / environments have been considered including Pure data <http://www.puredata.org> and Max/MSP <http://www.cycling74.com>, for developing a proof-of-concept RMA prototype. Due to licence cost overheads, Pd was preferred over Max/MSP and a first proof-of-concept patch for RMA has been developed using Pd. It is planned to consider another audio programming language "ChucK" <http://chuck.cs.princeton.edu/> for the first RMA release version. Investigations will focus on ChucK's support for integration with other languages and environments as well as performance evaluation and comparisons.



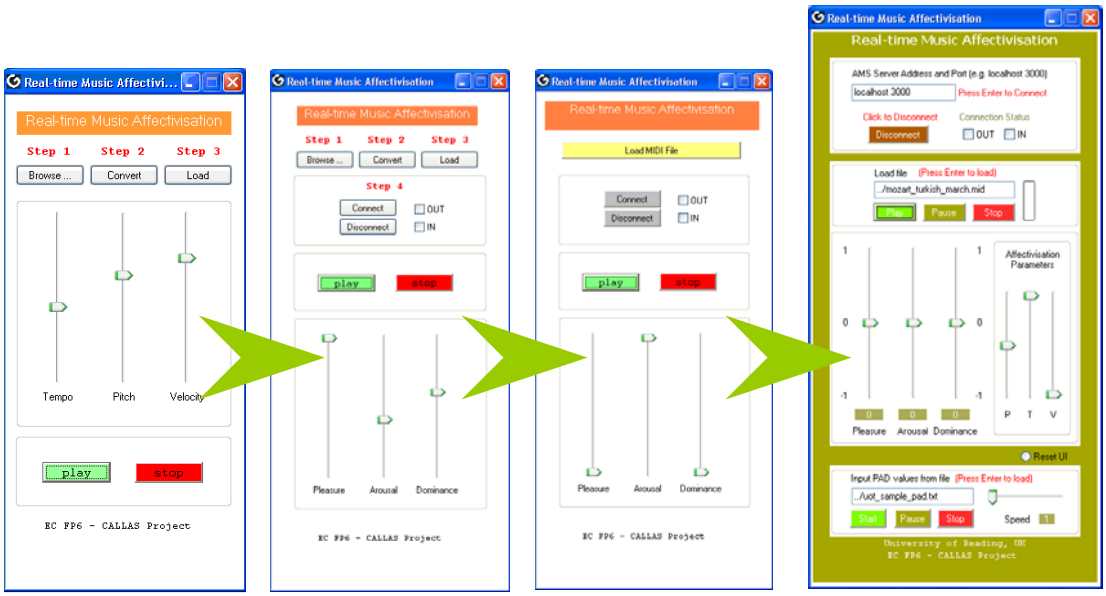


Figure 21 i) Real-time Music Affectivisation ii) Evolution (v1.2 to v1.5)

10.2 RMA proof-of-concept prototype in Pure data

The input for this proof-of-concept Pd patch is a MIDI sequence and a stream of Pleasure-Arousal-Dominance (PAD) values. Currently, these values are set manually using sliders however the sliders are able to receive data over TCP which may prove useful for integration with emotive input modules and components of the CALLAS project. In order to effectively use these PAD values for affectivisation purpose, the RMA Pd Patch connects to the AMS server over TCP and uses the real-time affectivisation service provided by the AMS Server. As part of this service, corresponding affectivisation parameters (pitch, tempo and velocity) are computed by the AMS Server from the client/user's Pleasure-Arousal-Dominance (PAD) model using the Case Based Repository (CBR) and these parametric values are then sent back to the RMA Pd Patch over TCP. Two separate ports are used for incoming and outgoing data. The patch gives as output affectivised MIDI sequence.

10.2.1 Version history

1. AMS-real-time-v1.pd:
    a. First proof-of-concept RMA Pd patch
2. AMS-real-time-v1-2.pd (with rma.gpd):
    a. First version of proof-of-concept RMA Pd patch with a GUI developed using GrIPD: Graphical Interface for Pure Data (GNU General Public Licence) <http://crca.ucsd.edu/~jsarlo/gripd/>
    b. Input: MIDI sequence and affectivisation parameters
    c. Output: Affectivised MIDI sequence
3. AMS-real-time-v1-3.pd (with rma-v1-3.gpd):
    a. Input: MIDI sequence and PAD model
        1. Patch connects to AMS TCP Server for using the real-time online PAD-model-to-affectivisation-parameters service
        2. Affectivisation parameters computed from PAD model by AMS TCP server and sent back to the patch
    b. Output: Affectivised MIDI sequence
4. AMS-real-time-v1-4.pd (with rma-v1-4.gpd):
    a. Updates and Changes:

1. Resolved the following issues by using xeq object for MIDI stream real-time manipulation
    1. Undesirable slight MIDI note timing alteration
    2. Browse, Convert and Load MIDI file before playing

5. **AMS-real-time-v1-5.pd (with rma-v1-5.gpd):**
    a. Current version of proof-of-concept RMA Pd patch with a GrIPD GUI.
    b. Updates and Changes:
        1. Resolved the following issues
            1. Specify Server IP address and port directly from GUI
            2. Select and load MIDI file directly from GUI
            3. MIDI output device selected at Pd start-up
            4. GUI automatically loads at patch start-up
        2. GUI updates
        3. Insert PAD values stream from file
            1. Load from GUI
            2. Control PAD value stream feed speed
        4. Display corresponding affectivisation parameter values processed and sent back by AMS TCP Server

10.2.2 Software and Hardware Requirements

1. Pure data (Pd) <http://puredata.info/downloads>
2. Graphical Interface for Pure Data (GrIPD) <http://crca.ucsd.edu/~jsarlo/gripd/>
3. MIDI output device / support

10.2.3 Installation and Usage

1. Download and install Pd
2. Download GrIPD from URL <http://crca.ucsd.edu/~jsarlo/gripd/#downloads> and follow installation instructions given below and/or at URL <http://crca.ucsd.edu/~jsarlo/gripd/#installation>.
    a. Unzip contents of gripd-*.zip to %Program Files%\pd\gripd\
    b. Go to folder %Program Files%\pd\gripd and copy gripd.dll to a path which is included in startup or path options of Pd e.g. %Program Files%\pd\extra\cyclone
        i. Alternatively, %Program Files%\pd\gripd (the path where gripd.dll is originally located) can be specified under path or startup options of Pd
    c. If the DLL is visible to Pd, it will display GrIPD startup information in the main window of Pd confirming that Pd has successfully found gripd.dll
        i. If this is not the case, it means Pd has been unable to load gripd.dll and hence the GrIPD-based GUI file for the Pd patch will not be loaded either
3. Copy rma-v1-5.gpd to %ProgramFiles%\pd so that when the patch AMS-real-time-v1-5.pd is run, the gripd object will be able to find the GrIPD GUI file (rma-v1-5.gpd) for RMA. Default path for gripd is the folder %ProgramFiles%\pd
4. Run the patch AMS-real-time-v1-5.pd

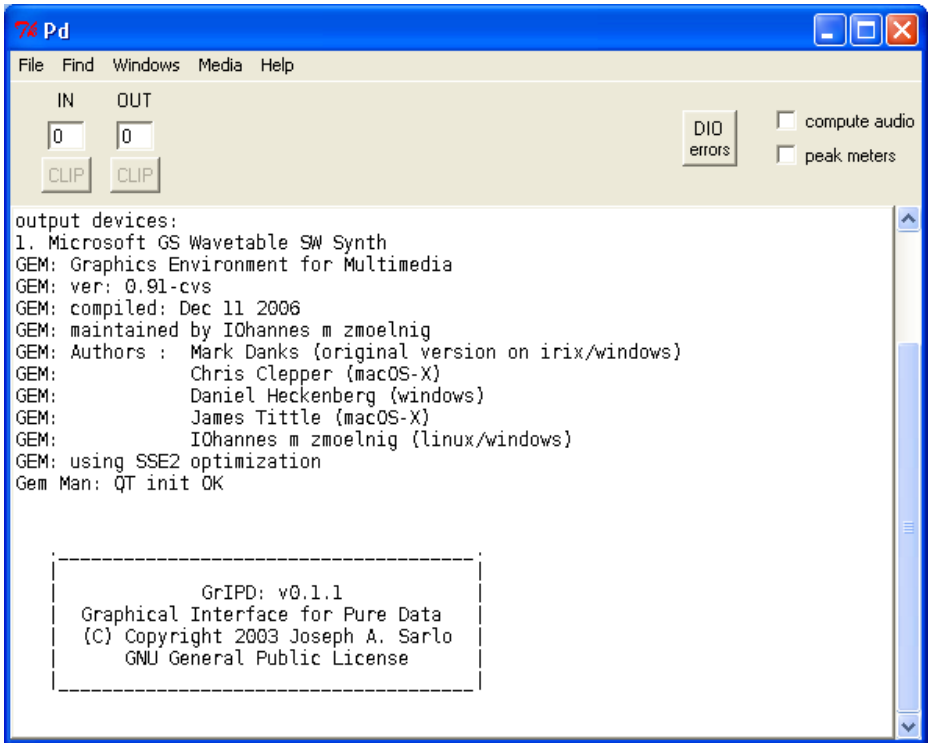


Figure 22 Pure data - main window

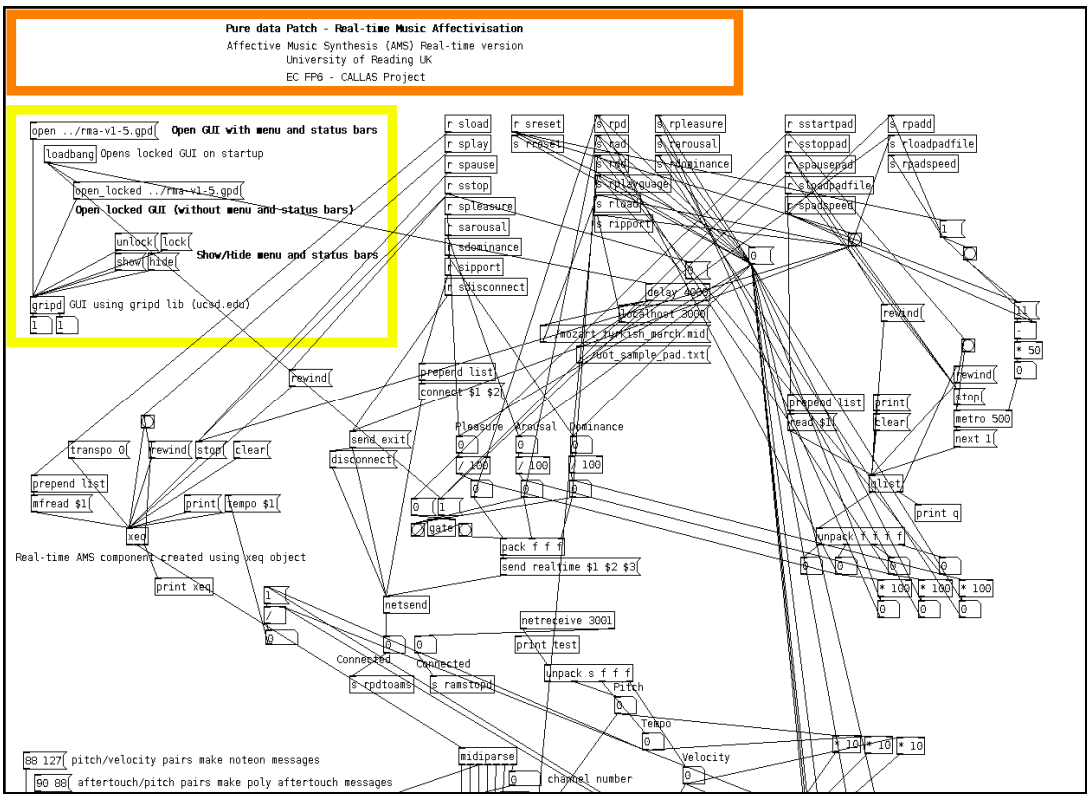


Figure 23 AMS – Real-time Music Affectivisation patch in Pure data

5. Under Media menu, select the option MIDI settings… and select an appropriate MIDI output device that is available on your machine as shown in figures below:

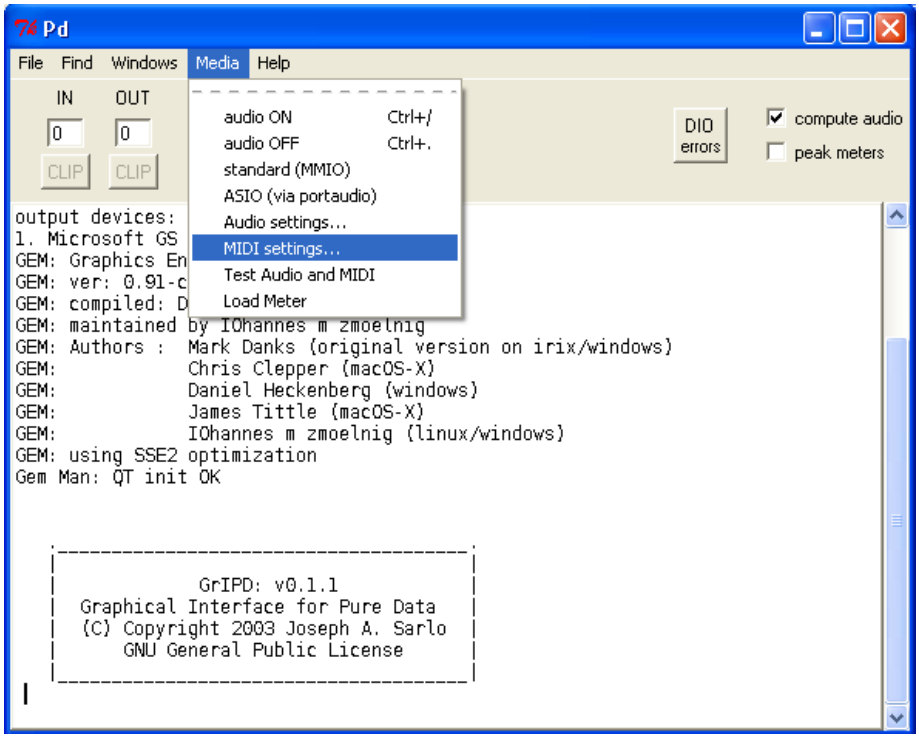


Figure 24 Pure data - MIDI settings

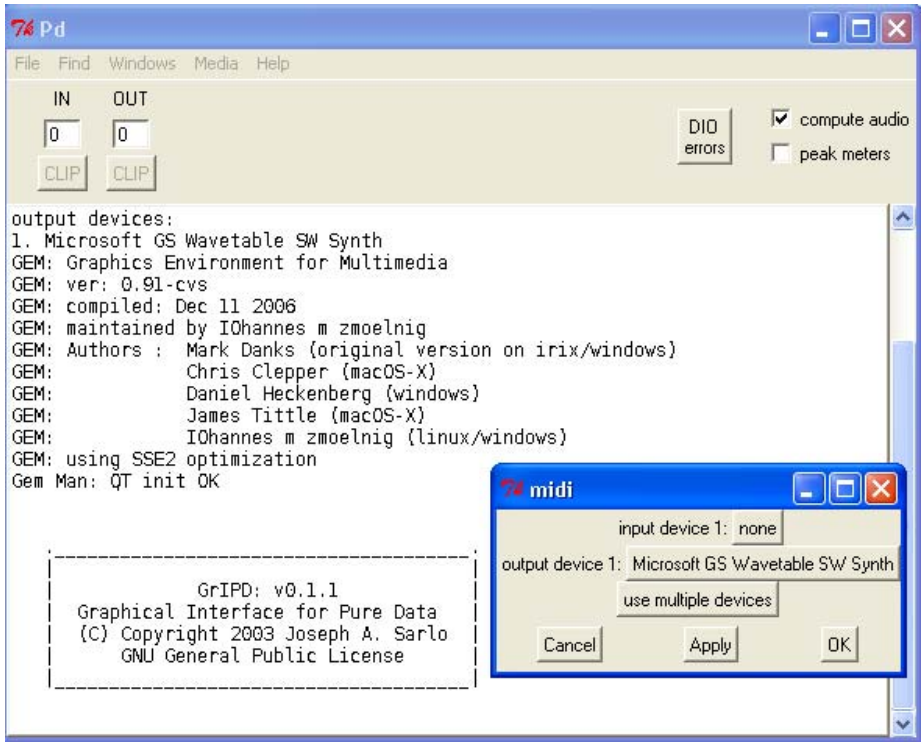


Figure 25 Pure data - set MIDI output device

6. If the GUI does not automatically launch, use the GUI controls in the patch to open GUI window in locked or unlocked mode.

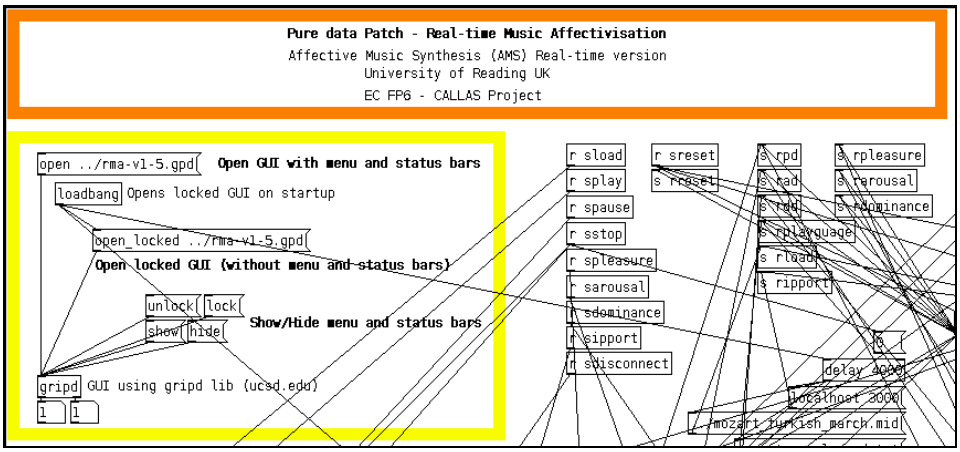


Figure 26 AMS - Real-time Music Affectivisation Pd patch GUI controls

7. Now use the GUI controls in the patch to open GUI window in locked or unlocked mode.

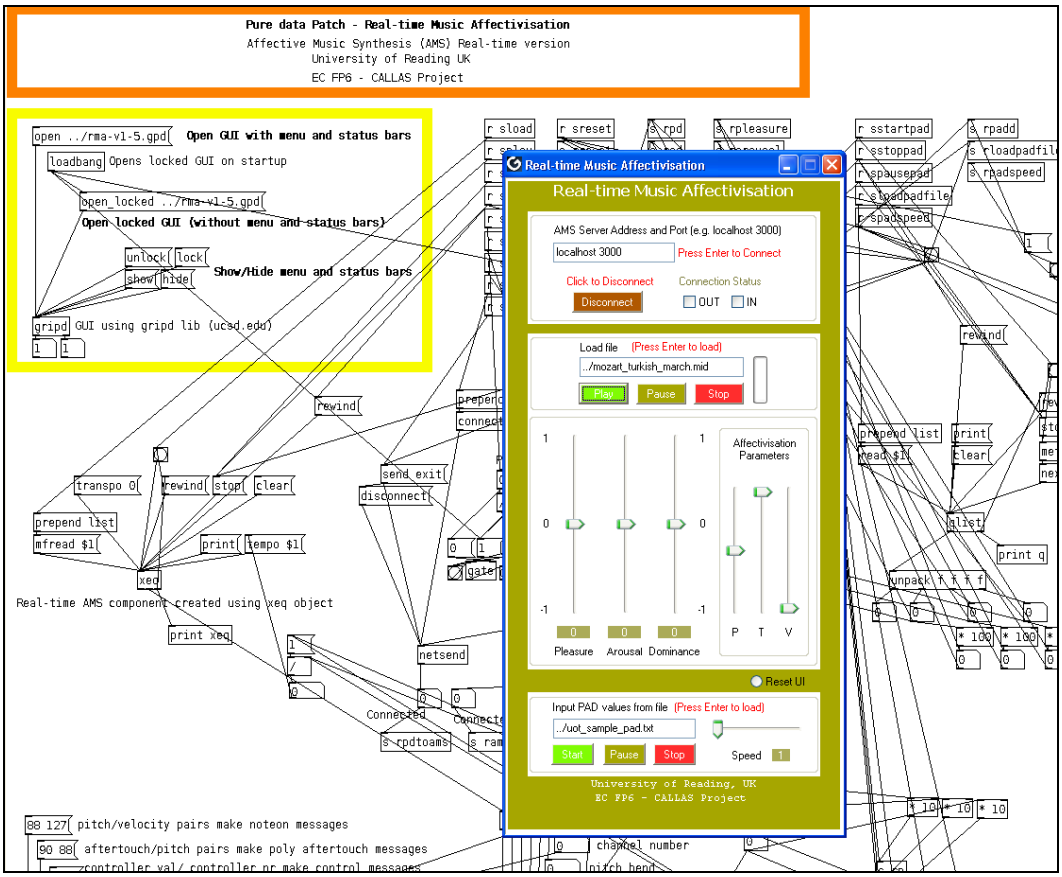


Figure 27 RMA GUI in GrIPD

8. Specify the IP address and port to connect to the AMS Server and press Enter.
    a. OUT and IN checkboxes will be checked automatically on successful connection
9. Specify the MIDI file to be played by using filename (if it resides in the same folder as the RMA Pd patch) or specify a path. Press Enter to load file.
10. Use play, pause and stop buttons to control playback of file. A vertical gauge displays playback by responding to velocity of notes played.
11. Use Pleasure, Arousal and Dominance sliders to hear real-time affectivisation by real-time and online computation of the affectivisation parameters (tempo, pitch and velocity).
    a. Slider values are displayed below each slider. These values are sent to the server for processing.
12. PAD values can also be supplied to the Pd patch via a text file in the following format:
    a. <time step> <P> <A> <D>; e.g. 100 0.83 -0.8 0.2;
    b. The speed slider in the "input PAD values from file" panel allows the user to control the rate at which PAD values are sent from the file to the server for affectivisation processing.
    c. Use play, pause and stop buttons to control playback of sample PAD values file.
13. The Affectivisation Parameters panel displays the pitch, tempo and velocity values corresponding to the PAD values processed by the AMS Server.
    a. Three sliders display these values in the panel. These values are sent back by the server post-processing.
14. Click Disconnect to disconnect from the AMS TCP Server.
    a. OUT and IN checkboxes will be unchecked automatically on successful disconnection

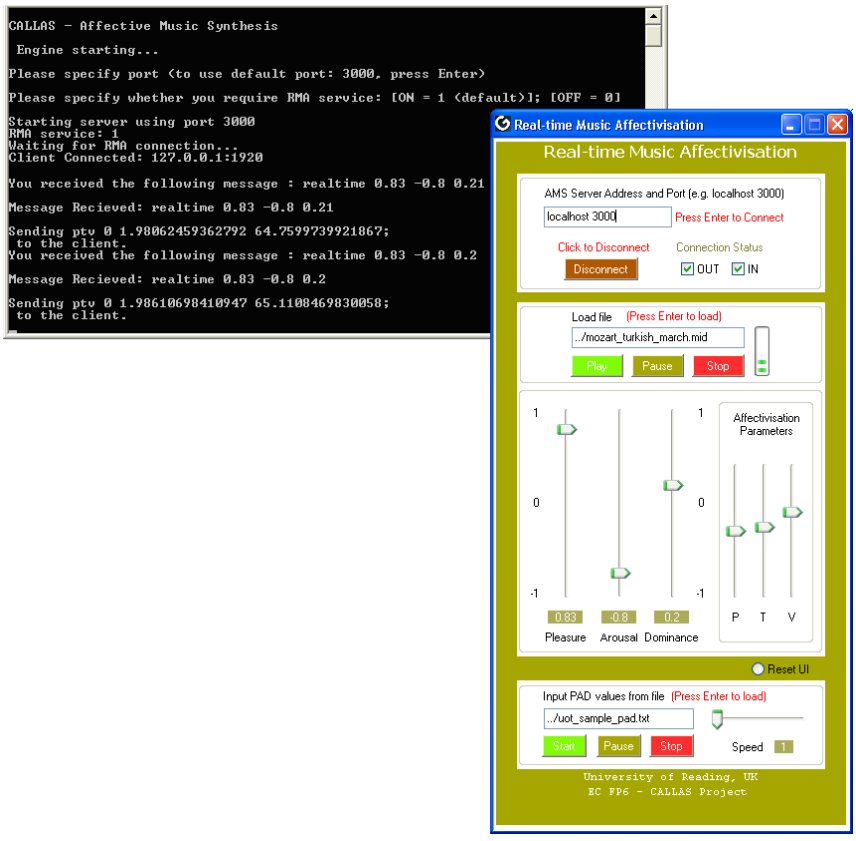


Figure 28 RMA Pd Patch GUI connected to AMS TCP Server

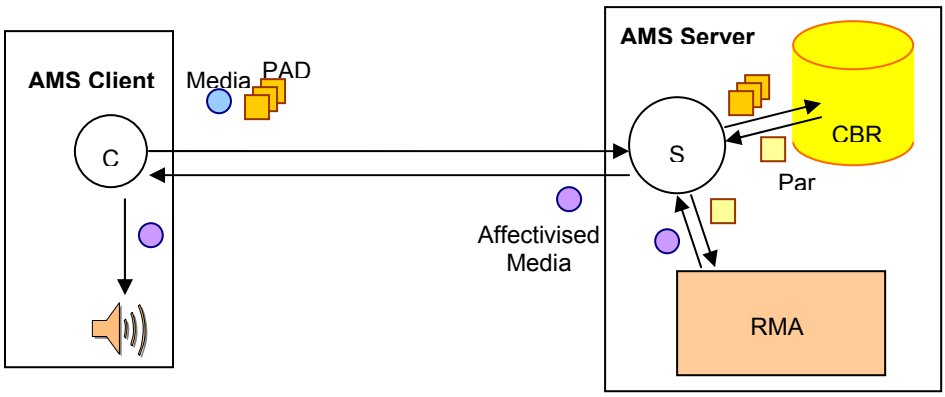
10.2.4 Important known problems

- None

10.3 Integration with AMS

The following two approaches were proposed for integration of the real-time music affectivisation patch/module with the existing Affective Music Synthesis (AMS) component. Currently, for RMA v1-3, Approach 2 has been used.

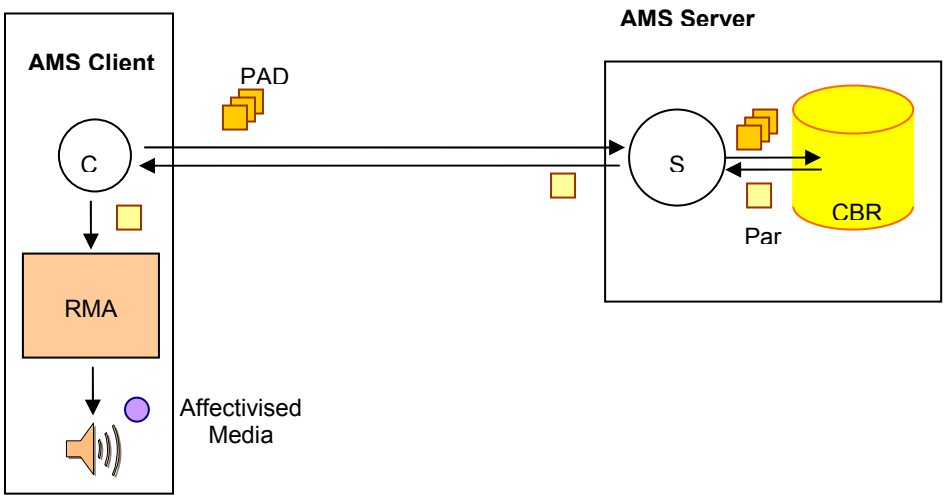
1. Client connects to AMS Engine, sends MIDI data and provides PAD values in real-time; Engine retrieves CBR matching values or processes new values (if no match exists) in real-time; Engine uses real-time patch for modification of MIDI sequence and progressively streams data to client for playback.
    a. Integrate the patch with existing AMS (To be done)



Server – Affective Music Synthesis TCP Server Engine
Client – Affective Music Synthesis TCP Client
CBR – Case Based Repository
PAD – Values for Pleasure-Arousal-Dominance Emotion Model
Media – MIDI to be affectivised
Par – Parameters of affectivisation i.e. tempo, pitch, velocity …
RMA – Real-time Music Affectivisation Patch

Figure 29: Approach 1 – Real-time Music Affectivisation patch at Server side

2. Client connects to AMS Engine and provides PAD values in real-time; Engine retrieves CBR matching values of affectivisation parameters or process new values (if no match exists) in real-time; Engine sends these values back in real-time to client; Client includes real-time music affectivisation logic, which uses these values to affectivise the MIDI sequence.



Server – Affective Music Synthesis TCP Server Engine
Client – Affective Music Synthesis TCP Client
CBR – Case Based Repository
PAD – Values for Pleasure-Arousal-Dominance Emotion Model
Media – MIDI to be affectivised
Par – Parameters of affectivisation i.e. tempo, pitch, velocity …
RMA – Real-time Music Affectivisation Patch

Figure 30: Approach 2 – Real-time Music Affectivisation patch at Client side

11. Conclusion and Future Directions

11.1 Current Status of AMS Prototypes

- Prototype of CALLAS Affective Music Synthesis (AMS) Module
    - Standalone version (delivered August 2007)
    - First TCP version using loopback (client and server on same machine, no file/content transfer to server side) (delivered December 2007)
    - Second TCP version (Client and Server on different machines), file/content transferred to server along with PAD values for affectivisation service (developed February 2008)
        - An improved/polished version (delivered October 2008)
    - Real-time Music Affectivisation (RMA) prototype patch developed in Pure data (Pd) with a GrIPD GUI v1.2 (delivered October 2008)
        - manipulates MIDI stream using affectivisation parametrics as direct input via sliders
    - Real-time Music Affectivisation (RMA) prototype patch developed in Pure data (Pd) with a GrIPD GUI v1.3 (delivered October 2008)
        - manipulates MIDI stream using PAD values as direct input via sliders
        - Connects to AMS Server
        - PAD model to affectivisation parameter values processed by AMS server and sent back to patch in real-time
        - MIDI manipulation performed by RMA Pd patch
    - Real-time Music Affectivisation (RMA) prototype patch developed in Pure data (Pd) with a GrIPD GUI v1.4 (delivered November 2008)
        - UPDATE: Resolved the following issues by using xeq object for MIDI stream real-time manipulation
            1. Undesirable slight MIDI note timing alteration
            2. Browse, Convert and Load MIDI file before playing
        - manipulates MIDI stream using PAD values as direct input via sliders
        - Connects to AMS Server
        - PAD model to affectivisation parameter values processed by AMS server and sent back to patch in real-time
        - MIDI manipulation performed by RMA Pd patch
    - Real-time Music Affectivisation (RMA) prototype patch developed in Pure data (Pd) with a GrIPD GUI v1.5 (delivered November 2008)
        - UPDATE: Resolved the following issues:
            1. Specify Server IP address and port directly from GUI
            2. Select and load MIDI file directly from GUI
            3. MIDI output device selected at Pd start-up
            4. GUI automatically loads at patch start-up
        - UPDATE: GUI updates
        - UPDATE: Insert PAD values stream from file:
            1. Load from GUI

2. Control PAD value stream feed speed
   - UPDATE: Display corresponding affectivisation parameter values processed and sent back by AMS TCP Server

- Agenda:
  - Integrate the non-real-time (or batch) version and the real-time version to develop a single next version configurable for online/offline, real-time/non-realtime use.
  - MPEG-SMR to enhance the parameters of affectivisation of music (currently three parameters provided by MIDI are used: pitch, velocity and tempo)

SVN structure

11.1.1..1 AMS TCP prototype version 2

Current version (v2) of the AMS TCP prototype can be located at CALLAS SVN (http://www.callas-lab.eu/svn) under the folder /ams

- Note: /ams/bin includes the standalone AMS version with necessary DLLs (Early prototype, non TCP)

Use CALLAS Affective Music Synthesis solution file to run solution with all necessary projects including demos for standalone AMS and TCP version with AMS server and AMS client.

- /ams/src/CALLAS_AMS_Demo includes src and bin for standalone AMS version

- /ams/src/CALLAS_AMS_TCP
  - includes src and bin for AMS TCP server
  - console based
  - listens for incoming connections
  - performs affectivisation of music piece sent by client

- /ams/src/CALLAS_AMS_Demo_TCP
  - includes src and bin for a demo TCP client
  - connects to AMS TCP Server
  - GUI
  - Sliders for setting PAD values manually (demo; replaced by PAD streams from CALLAS emotive input components)
  - Output sliders displaying corresponding affectivisation parametric values (computed by AMS TCP Server)
  - Two-way File transfer (music piece sent by client to server for affectivisation; affectivised piece sent back to client)

11.1.1..2 AMS real-time prototype a.k.a Real-time Music Affectivisation RMA version 1.5:

Download from Wiki:
http://wiki.callas-newmedia.eu/twiki/bin/viewfile/Main/AffectiveMusicSyntesis?rev=1;filename=CALLAS-AMS-RMA-v1-5.zip

Includes:

- RMA Pd Patch (proof-of-concept) .Pd file (v1.5)
- RMA Patch GUI (GrIPD) .gpd file (rma-v1-5.gpd)

- README
- RMA documentation .doc file (Affective Music Synthesis RMA v1-5)
- CALLAS AMS TCP Server binaries

Version 1.5 contains GUI updates and issue resolution reported in previous sections.

Version 1.4 updates included use of xeq object to manipulate MIDI streams and requesting the AMS TCP server for PAD to affectivisation parametric processing on-the-fly. The GUI (rma-v1-4.gpd and above) provides controls for connection/disconnection to/from the AMS server and sliders to manually input PAD values in place of tempo, pitch and velocity sliders (as in v1-2 and v1-3).

11.2 Mapping PAD Emotional Model to Affectivisation Parameters

An interesting feature for the Affective Music Synthesis module could be to use alternating emotional models, one mapped directly proportional and the other inversely proportional to the emotive input, such that the directly proportional mapping would drive the AMS module to suit a user's "Relaxed" mood, whereas an inversely proportional mapping would bring the user's mood from "Anxious" to "Relaxed", for instance.

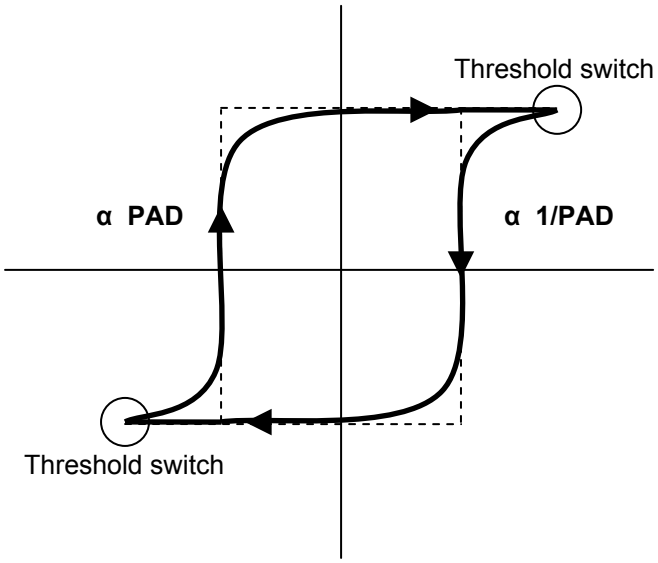


Figure 31: Hysteresis loop for the twp mappings of PAD Emotional Model with Affectivisation Parameters

A threshold switch for alternating between the two mappings could either be statically provided (worst case scenario) or be dynamically computed by maintaining a history of user moods and affectivisation services rendered to use in a pre-specified time window.

This threshold switch could also be acquired as input to the Affective Music Synthesis module from the user directly, e.g. in case of volume as an affectivisation parameter, the power of the audio output at user side could be used to determine the volume level at which auditory experience starts to overshoot a recommended safe decibel range. However, this additional input parameter would a) require a profile management system be incorporated for maintenance of up-to-date information on user and user's device etc. and b) not completely fall in line with the concept of CALLAS Shelf (i.e. all CALLAS modules take as input PAD values only).

11.3 Affective Music Synthesis Improviser

Currently, the AMS module affectivises and modifies the music piece in context of the emotive input representing the user's mood. An Affective Music Synthesis Improviser Add-on could be developed to synthesise music using a repository of short loop-able samples such as beats, rhythms, melodies etc.

A look-up table could determine the time signature to suit a particular user mood. Such a module would piece various loops together to form a short piece of music to be played back in a loop.

If this piece is MIDI based, it could be streamed via real-time patch to affectivise it based on incoming PAD values. Together with the AMS Improviser Add-on and the real-time music affectivisation patch (RMA), AMS would be able to remove / add / update new patterns on to the synthesised music as the mood changes from one state to another. For example, if a mood change requires the tempo to be increased, at a computed threshold, the improviser could decide to update the beat pattern being used. An illustration is given in figure above.

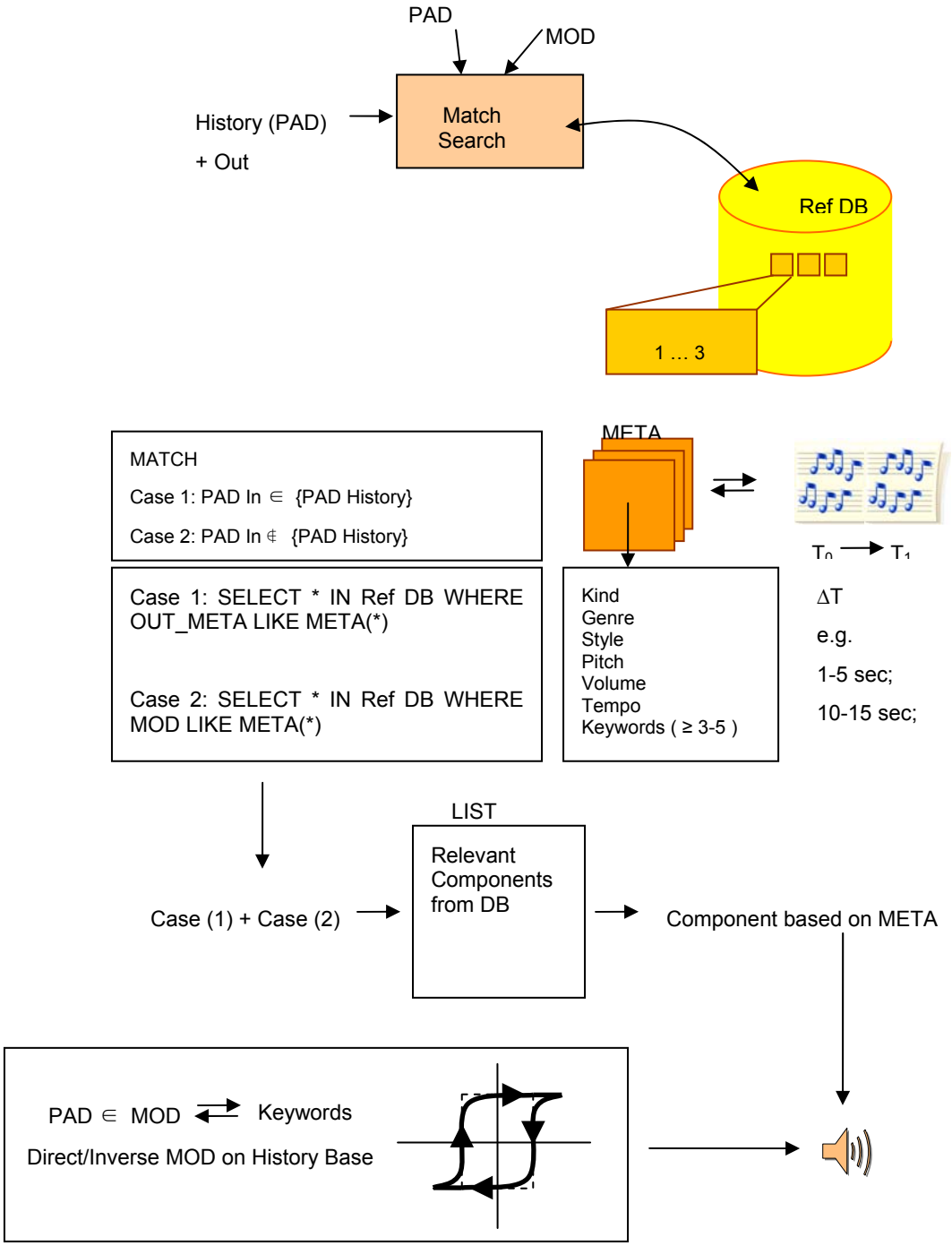


Figure 32: Affective Music Synthesis Improviser

11.4 Routes

11.4.1 Route I

- Two types of mappings between the PAD model and the affectivisation parameters set (directly proportional and inversely proportional)

- Step by Step offline affectivisation (non-real time), by switching between the two mappings
- User would only perceive difference after each playback of affectivised music piece
- A playlist feature could be introduced to enhance user affectivisation perception
- Affective Music Synthesis Improviser

11.4.2 Route II

- Two types of mappings between the PAD model and the affectivisation parameters set (directly proportional and inversely proportional)
- Immediate real-time affectivisation of MIDI stream switching between aforementioned mappings
    - Requires a real time affectivisation module (RMA Pd prototype patch already developed)
- User would perceive difference/response immediately
- Affective Music Synthesis Improviser

11.5 Conclusion

Since:

- the core components of AMS (CBR etc.)
- a non-real-time affectivisation prototype a.k.a. batch or offline version of AMS using toub.midi library; and
- a real-time music affectivisation prototype (AMS-RMA) using Pd and GrIPD

have already been developed. We intend to follow the aforementioned **Route II** (superior in terms of quality of experience due to real-time affectivisation). Therefore, the following specifies the future directions:

- Development of two types of PAD-model-to-affectivisation-parameters mappings (directly proportional and inversely proportional) including a trigger switch
- Development of Affective Music Synthesis Improviser to create affective music from a repository of pre-recorder short loopable samples

References

[1] CBR in Context: The Present and Future, *David B. Leake*, (accessed 23/04/2007), Available in WWW: http://www.cs.indiana.edu/~leake/papers/p-96-01_dir.html/paper.html

[2] Getting Caspian, *University of Wales, Aberysywyth*, (accessed 23/04/2007) Available on WWW: <http://www.aber.ac.uk/compsci/Research/mbsg/cbrprojects/getting_caspian.shtml>

[3] Inside Case Based Reasoning, *Christopher K. Riesbeck*, Lawrence Erlbaum Associates Inc.

[4] Midi for .NET, *GotDotNet*, Available on WWW: <http://www.gotdotnet.com/community/usersamples/details.aspx?sampleguid=89cde290-5580-40bf-90d2-5754b2e8137c> (accessed 31/08/2007)

[5] The PAD Comprehensive Emotion (Affect, Feeling) Tests, *Albert Mehrabian*, (accessed 24/06/2007) Available on WWW: <http://www.kaaj.com/psych/scales/emotion.html>

[6] Personality Tests: A General Description of Temperament, *Albert Mehrabian*, (accessed 24/06/2007), Available on WWW: <http://www.kaaj.com/psych/scales/temp.html>

[7] SaxEx: a case based reasoning system for generating expressive musical performances, *Josep Lluís Arcos, Ramon López de Mántaras, Xavier Serra,* 1998

[8] NOOS, *IIIA Spain*, (accessed 12/07/2007) http://www.iiia.csic.es/Projects/NOOS.html

[9] Getting Caspian, *University of Wales, Aberysywyth*, (accessed 23/04/2007) Available on WWW: <http://www.aber.ac.uk/compsci/Research/mbsg/cbrprojects/getting_caspian.shtml>

[10] Case Based Reasoning in CALLAS, IMSS University of Reading, CALLAS Project, May 2007

[11] The PAD Emotional Model, IMSS University of Reading, CALLAS Project, July 2007

[12] CALLAS Progress Report – July 07, IMSS University of Reading, CALLAS Project, July 2007

[13] Implementation Documentation, IMSS University of Reading, CALLAS Project, August 2007

[14] Demonstration Program User Manual, IMSS University of Reading, CALLAS Project, August 2007

[15] Using Background Music to Affect the Behavior of Supermarket Shoppers Ronald E. Milliman *Journal of Marketing*, Vol. 46, No. 3 (Summer, 1982), pp. 86-91 doi:10.2307/1251706. Available on WWW: <http://links.jstor.org/sici?sici=0022-2429(198222)46%3A3%3C86%3AUBMTAT%3E2.0.CO%3B2-7>

[16] Grachten, M. (2001) JIG: Jazz Improvisation Generator. *In Proceedings of the MOSART Workshop on Current Research Directions in Computer Music, 1-6.* Barcelona, Spain: Pompeu Fabra University Publishers

[17] Composing Music with Case Based Reasoning (SICOM) Pereira, F. C. , Grilo, C. , Macedo, L. , Cardoso, A. <http://www.cisuc.uc.pt/view_pub.php?id_p=67>

[18] Pd – Pure Data. Available on WWW: <http://puredata.info/>

[19] Max/MSP – Cycling 74. Available on WWW: http://www.cycling74.com/

[20] Rosalind W. Picard, Elias Vyzas, and Jennifer Healey (2001) Toward Machine Emotional Intelligence: Analysis of Affective Physiological State Ieee Transactions On Pattern Analysis And Machine Intelligence, Vol. 23, No. 10

[21] Jae-woo Chung, G. Scott Vercoe: The affective remixer: personalized music arranging. CHI Extended Abstracts 2006: 393-398

[22] Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. Current Psycho., 14(4):261--292, Dec. 1996
[23] Personality tests: A general description of temperament. Available on WWW: http://www.kaaj.com/psych/scales/temp.html
[24] CALLAS Framework. Available on WWW: <http://www.callas-newmedia.eu/the-framework>
[25] MPEG Group on SMR Symbolic Music Representation. Available on WWW: <http://www.interactivemusicnetwork.org/mpeg-ahg/>
[26] PAD Temperament. Available on WWW: <http://www.kaaj.com/psych/scales/padsoft.html>