New Strategies for Computer-Assisted Composition Software: A Perspective

Kevin Dahan[1,2], Guy J. Brown[1] and Barry Eaglestone[2]

Department of Computer Science[1] and Department of Information Studies[2],
University of Sheffield, Regent Court, Sheffield S1 4DP, U.K.
email: kevin.dahan@wanadoo.fr

Abstract

The most prominent problem of designing a computer-assisted composition system lies in the fact that composers do not have a common way of expressing themselves. Hence there is a tension between the composers themselves and the software they use, which usually relies on a specific grammar and may implicitly assume a specific way of composing. Moreover, the many formats for music representation that can be used in a composition environment produce another difficulty for the designer, who must deal with a large number of heterogeneous and unpredictable formats. We believe that computer-assisted composition systems can overcome these difficulties by placing more emphasis on rich representations of electroacoustic music. An object architecture for achieving this is described.

1 Introduction

Computer music composition systems aim to offer composers a convenient way of expressing themselves, allowing them to manipulate heterogeneous data types (such as MIDI, audio, common music notation, etc.) and arrange them in a structured way. However, provision of fertile environments for creative users is a more elusive problem, and has been largely unresearched.

In this paper we report on progress in research, within which we have taken a holistic view of composition systems. We are addressing both above aspects of composition systems, through seeking a better understanding of the ways in which composers interact with computers when they are being creative. The research is grounded in a preliminary qualitative analysis of composers at work (Eaglestone *et al.* 2001) (Nuhn *et al.* 2002) in which we have gained a new insight into composers' needs. The main contribution of this paper is a novel composition system architecture, which directly addresses the results of our previous study of composers at work.

The paper is organized as follows. Section 2 provides a prelude to our discussion of composition software design issues, by briefly summarising the notions that have emerged from this study. In Section 3, we then discuss the implications of those notions with respect to composition systems software. In Section 4, we propose an architecture that directly addresses the conclusions and describe aspects of our prototype, and finally Section 5 details the current prototype's implementation.

2 Why do Composers have problems with current systems?

In this section we establish the motivation and research base for our prototype, by summarising findings of our study of composers at work (more extensively covered in (Nuhn *et al.*, 2002) in which we have explored inherent tensions that appear to exist between composition systems and the requirements of composers. Our empirical study involved observing in depth a small number of electroacoustic music composers at work, and followed the naturalistic paradigm, described by Lincoln and Guba (1985). Rich data was collected through experiments in which composers were commissioned to work on pieces over protracted periods, using familiar software and hardware. Data was then analysed using qualitative research methods (Eaglestone *et al.*, 2001), and a grounded theory approach (Ellis, 1993) in particular.

Eleven main notions emerged from this study: (i) Composers are a heterogeneous user group, with different perspectives on the problem domain and fundamental differences relating to the dichotomy between creating music for the music's sake and creating music as a showcase for new tools and techniques. (ii) Diversity of composition software tools has a positive impact on creativity. (iii) Differences between the users' perceptions and the semantics of information representations within the composition systems have a positive impact on creativity. Systems that challenge a composer's preconceptions and present the unexpected are particularly valued. (iv) Graphical user interfaces (GUIs) are often too rigid to express those associations between artifacts which are important to a user. (v) Spatio / visual processing could serve as an aid to creativity. Within music, this extended to

processing based upon body movement, which is a natural expression of musical ideas. (vi) Vision plays a major role in the creation of sounds, even the look and feel of the software interfaces, as well as representations of the sounds themselves. However, visual representations can have a negative impact by providing misleading impressions about the actual quality of sounds. This is recognised by some composers who will frequently listen to sounds without any visual representations. (vii) Touch plays a major role in the creation of sounds, and there seems to be a general desire to physically touch the sounds, implying the need for force feedback interfaces. The need for physically engaging with the tools composers are working with feeds our overall impression that all composers have to cope with the distance between physical and virtual domains. (viii) Principles important to the user may be hidden by interfaces that operate at too high a level of abstraction. Availability of easy-to-use, heavily destructive (real-time) processing tools and the easiness to assemble a huge number of sounds over a short period of time seems to create situations where composers are no longer aware of the processes involved in their sound manipulation. This can lead to over-processing and over clustering of sounds, resulting in music which does not refer to any common, shared experience but the experience within the composing community. (ix) There is a role for both approximate real-time manipulations and non-real time manipulations which provide greater accuracy. (x) Increased knowledge exchange and a "know-how" data base are likely to be advantageous. (xi) Much creativity happens away from the computer. Even short interruptions can act as a huge inspiration.

These notions may be explained more generically, in terms of models of creativity as divergent thought and the "flash of inspiration", often as a result of serendipity (Guilford 1967) (de Bono 1987). The first three notions can be generalised as references to: the idiosyncratic nature of creative users (notion (i)); the "voyage of discovery" nature of the search for creativity solutions (ii); and hence the need for computer systems that challenge, rather than reflect, users' preconceptions to catalyse creativity (iii). Thus, the user should be allowed to make free associations between artifacts, unconstrained by assumptions built into the user interfaces (iv). Discovery of these associations may be aided by spatio / visual processing (v). In general, there is a need for interfaces, which allow the creative user to immerse in the problem and the materials from which the solutions may be constructed (vi and vii) and to override semantics and assumptions built into the high-level interfaces they use (viii). The latter will enable users to reconfigure the problem and solutions from basic components, i.e., a Gestalt approach (Reybrouck 1997). Systems should support both rapid intuitive exploration of ideas and also their detailed refinement and elaboration (ix). Explicit support for individual and community knowledge is important (x). The time away from the computer is also important, because of limitations of current software systems, but also makes possible aspects of creativity in which the computer has no role (xi).

These observations form the grounding for our current research, within which we are prototyping a composition system to test and expand the theories developed in our initial analysis.

3 How Can Composition Systems Be Improved?

We now consider some implications of the above findings, as they relate to composition systems, as the next step in our argument towards the composition system architecture proposed in sections 4 and 5.

Firstly, we note that heterogeneity of the user population, the diversity of composition software tools they require, and the experimental nature of the area, all mediate against a single generic "standard" software solution. Therefore our aim is towards general design principles for composition systems.

The notions in Section 2 have implications for conventional approaches to system design. These stem from a fundamental difference between composers and users of "conventional" software systems. The latter create working environments with logical structures, constraints and processes that reflect accepted perceptions of problem space semantics and thus restrict the users to working within constraints of "good practice". Such systems are therefore well suited to prescribed tasks, such as business processes, and computer-aided engineering and design applications. In contrast, composers are creative users, who often choose to work with multiple software tools at all levels of abstraction and which challenge, rather than reflect, their perceptions. The experimental and creative nature of electroacoustic composition determines that composers constantly seek new forms of musical expression and so the notion of work within the bounds of "good practice" is inappropriate. Further, the importance of randomness, serendipity and capitalising upon the by-products of mistakes set creative users apart.

The scenario of composition systems that host diverse collections of composition tools has similarities to conventional distributed systems, such as multidatabase systems (Bukhres & Elmagarmid 1996). Both aim to provide access to multiple heterogeneous systems. Consequently each subsystem independently defines its own services and data resources, and hence its own architecture. However, in order to expose composers to the diversity and idiosyncrasies of the services, representations and interfaces of the hosted tools, a composition system must *aggregate* them, whereas conventional approaches aim to *integrate* resources of

the component system. A composition system therefore must provide access to the tools it hosts in a semantically neutral manner, so as not to mask from the users the characteristics and idiosyncrasies of each component system.

The experimental nature of electroacoustic music composition necessitates composition systems which are open with respect to the services they support. Composers should be able to import new and unfamiliar software tools, and possibly modify existing tools and create new ones. This contrasts with conventional systems in which services are predefined and/or defined and maintained by the administrator rather than by individual users.

The "voyage of discovery" nature of composition and consequential importance of serendipity and randomness conflicts with the notion of the "external model". Conventionally, a user interface is based upon an external model which provides users and applications with relevant data and services in forms that are meaningful to them. However, this facility is best suited to convergent thought whereby ideas are refined and elaborated, since an external model predetermines and constrains the scope and structure of the information accessible to users. Consequently, potential for randomness and serendipity as catalysts to creativity and the ability to freely make associations between artifacts are restricted. Therefore we propose a system interface based upon an external model without predefined contents, constraints and semantics, so as to better support creative behaviour of composers, since this will allow them freely and dynamically create their own external models to describe the structure and semantics relating to a composition as it takes shape.

The observed importance of "time away" from the computer system during composition further supports this proposal for an initially null external model, since time away is often used to find alternative, possibly less formal, forms of design facility not provided in the system being used. This suggests it would be beneficial for composition software to provide capabilities for informal writing and sketching, to create both data and metadata. (This is in line with Puckett's proposal (Puckette 2002) that composition systems should provide facilities for describing compositions at an abstract level, with the same expressive possibilities as a pen and paper musical score). A composer can then selectively assign formal semantics to parts of the sketched artifacts by delimiting and mapping them to objects and types defined within the system.

Apparent tension between associations that are important to composers and those that can be expressed on the GUIs of current systems further supports the notion of a workspace that can be dynamically defined by the user as the composition evolves. This workspace should allow arbitrary associations between artifacts, thus making it possible to represent what is perceptually important.

Our observations suggest the need for further research into HCI aspects, specifically focusing on holistic approaches to visual representations of audio information, including macro and micro levels of detail, so as to better communicate properties and quality of music artifacts. It has been noted that early electroacoustic composers used graphical scores preceding or accompanying the music, and it is well known that many current electroacoustic composers rely on graphic schemas before and during the time of composition (Budon 2000). Moreover, at the micro level, composers of electroacoustic music are concerned with audio waveforms and their characteristics (such as statistical properties) that relate to the listeners' perceptions of the sound, whereas at the macro level, they are concerned with the ways sounds are combined, i.e., musical structures. Useful micro and macro information can be automatically inferred from audio artifacts using known techniques. We therefore believe that it would be beneficial for a user-defined external model, as described above, to be complemented by metadata inferred by the system itself.

An architectural implication of the above discussion of HCI issues is that, rather than accessing the database in the conventional manner, i.e., via the restricted and personalized view provided by an external model, the user must have access to audio processing functionality at each level of abstraction, ranging from perceptual representations to the acoustic waveforms and signal processing algorithms. This suggests that the interface should be "side on" to the architecture, ranging over all the levels.

4 A First Model of the System's Architecture

In this and the following section, we provide a snapshot of the prototype we are developing to test the validity of the above analysis. This section presents the conceptual architecture devised for the prototype and discusses issues relating to the design of the prototype. This is preliminary to an outline of the object model for the system that is being implemented in Section 5.

4.1 Conceptual System Architecture

An outline of the architecture of the prototype composition system is given in Figure 1.

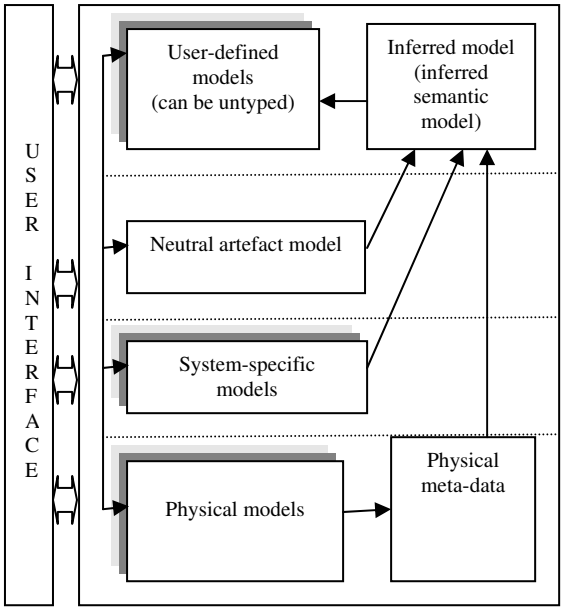

Figure 1: Outline architecture for an exploratory prototype composition system.

This architecture is a very literal implementation of the above guidelines. The system-specific models represent services, structure and constraints of the various tools hosted by the composition system. The physical models represent how the data, such as sound files and source files, are physically stored. The neutral artifact model aggregates by providing a directory of the musical artifacts created using the hosted tools. The user-defined models provide the workspaces within which composers can dynamically assemble the semantics associated with their composition(s). Specifically, support for tablet input from within the prototype is provided to allow free-hand sketching and annotation of these models, with the facility of selectively assigning formal semantics to artifacts by delimiting them and mapping them to artifacts defined within the other models. This information is complemented by the inferred model and physical meta-model, which respectively contain semantic (macro) and statistical (micro) information inferred from the audio objects. The user interface allows access to all components.

4.2 Implementation issues

The realization of this architecture tackles, head on, two enduring problems in computer music; that of representing abstract musical structures, and working with multiple levels of abstraction.

Abstract musical structures

Representing data structures is a challenge in itself, but this holds particularly true when dealing with musical structures, in which the multiplicity of operating levels is enormous and "meaning" is always present (Vaggione 2001). An important problem in composition software is the lack of a fixed abstract representational system (such as notation in classical music), which provides users with the ability to work from global organization to low-level without the problem of changing/adapting their grammar. This issue has been addressed several times (Windsor 1995) (Puckette 2002), but is not yet widely accepted by composition systems designers.

In order to palliate to the lack of standard method in electroacoustic music "notation", the strategy followed by most software that provides a facility to handle this level of abstraction is to let users attach graphical objects to lower-level structures, hence "covering" the data with a metaphor that (in the composer's idiom) describe the behaviour and/or morphology of the underlying sound objects. This facility is present in PureData (Puckette 2002) (Figure 2) and to a lesser extent, resembles the mechanism of *maquette* provided by OpenMusic (Assayag, Baboni and Haddad 2001) (Figure 3).

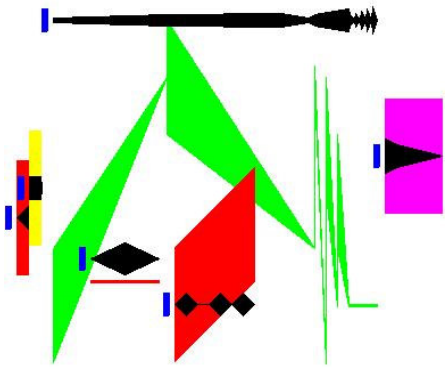

Figure 2: Symbolic representation in PureData

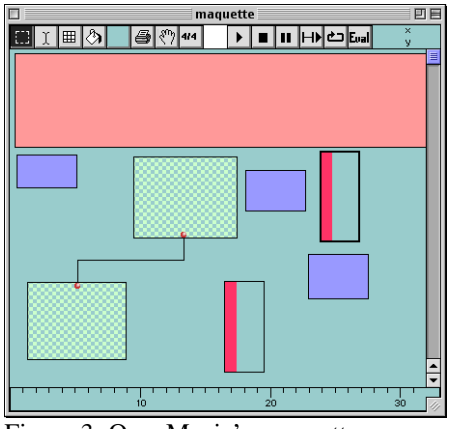

Figure 3: OpenMusic's *maquette*

This strategy is interesting, as it permits composers to work with concepts rather than physical data, but it does not solve the hiatus between the representational system and actual data, as it is merely a "cosmetic" layer – there is no real (as in "morphological") manipulation of data provided through this level of representation/abstraction.

In our previous study, several composers expressed frustration that software did not provide

guidelines or even symbolic representations *related to sound structures* during the composition process (i.e. visually providing a meaning that contrasted and/or conflicted with what they experienced by hearing the sound). "Augmented" representations, in which a visualisation of statistical properties is added to the more conventional representation, can be a solution to palliate to this feeling.

Defining graphical and/or complex objects with multiple levels of representation is ineffective if there are no underlying mechanisms which allow users to manipulating data directly from any levels. Though it is unrealistic to imagine the system "understanding" a symbolic manipulation, the software should allow a composer to describe and use her own idiom and the software should react appropriately. This is unlikely to be considered a problem for composers, since "serendipity and randomness are highly valued" (Nuhn *et al.* 2002).

Further, we suggest that a mechanism is needed which can collect and store statistical data from observations of the user's behaviour, i.e., a learning mechanism. Ultimately, this would permit the system to infer models of composers' working methods.

Working with Multiple Level of Abstraction

Direct access to the different levels constituting a musical structure is important. To achieve this, we must allow users to choose at which level(s) of representation/abstraction they will work on a given object, and draw relations between objects at different abstraction levels.

We decided to employ the usual triadic separation commonly used in semiotics to distinguish between levels of abstraction: physical, statistical and abstract levels corresponding to the classical objective, subjective and semiotic levels of the object.

The *physical level* can be regarded as one in which there is minimal interpretation or deformation of data. At this level, we try not to hide any information – hence the representation is both complex and difficult to manipulate directly. This level embodies a number of representation methods. For example, audio signals can be represented as waveform, Fourier spectrum, wavelet transform, and so on, and all should be available to the user.

The *statistical level* is perhaps the more interesting level at which composers expressed their wish to experiment (Nuhn *et al.* 2002). Representations at this level are collections of statistical data over the signal (such as mean pitch, maximum event duration, etc.). This allows the user to have direct access to the global organization of musical structures, and manipulate high level functions without getting into excessive details. This kind of manipulation of sound using stochastic and statistical methods is common in granular synthesis algorithms (Roads 1998), but is only sparingly used in more general computer music composition software dealing with heterogeneous data structures.

The *abstract level* of representation is the most complex level to implement (and to describe), as it is unpredictable by definition. Abstract representation can consist of a variety of "untyped" formats, such as text files, drawings, and fragments of notation (explicitly not interpreted by the system). We propose a mechanism, similar to the technique of information hiding we described in the previous section, but with the difference that our system does not impose a linkage to a semantically typed structure. This allows the user to create a meta-level of meaning which is only relevant to him.

Another feature proposed by our design is the possibility of the system proposing an abstract representation, based on inferences it produces using statistical and physical representation. This produces a meta-representation which is intended to embody the different aspects of a sound or a musical phrase, hence providing a bridge between micro- and macro-levels of abstraction and representation. This is something composers have long sought (this issue has been partly addressed by the use of self-similarity (Yadegari 1991) and/or genetic algorithms (Burton and Vladimirova 1999)).

Inter-levels Operations

A most important design problem associated with the above multilevel representation strategy is how to achieve propagation of a morphological modification at a high level through the lower levels of abstraction. The inverse relation (i.e. bottom to up) is more straightforward. Correlating complex structures that have relations expressed at various abstraction levels also poses a critical problem.

Our strategy for supporting this propagation is to rely on separate objects for each level (so-called "views" in object oriented methodology (Meyer 1997)). Such objects can be manipulated without harming the original signal and modification of the original signal (and, therefore, of the other views) is only effective when switching to another level of representation.

Representation of Heterogeneous Structures

Another powerful feature of using separate objects for each representation level is the ability to "transfer" qualities of one object to another of different type and/or representation level. For example, the mean loudness of a MIDI phrase can be modified and applied to a different MIDI phrase or sound file. This allows composers to create complex relations between musical structures, in a more flexible way than other software currently allows.

As each representation level is an object in itself (independent from the original signal), relations are not only expressed between "raw data" (i.e. the musical signal), but also between specific abstract levels of a musical signal – hence providing the possibility of creating complex linking between the different abstraction levels of the same musical artifact, possibly using external constraints. We

believe that such complex heterogeneous relationships are desired by composers, and provide them with the "challenge" they require from the software (Nuhn *et al.* 2002).

5 Implementation

This section explains the strategies and design logic we are currently following to develop our prototype composition system based on the strategy of trans-level communication previously described. The current prototype is implemented using Squeak Smalltalk, to facilitate rapid prototyping, but will be ported to Java at a later date.

We outline the two main designs on which the system is built, thus expanding the conceptual architecture shown in Figure 1.

5.1 Representational Model

One of our main goals was to allow users to choose the abstraction levels at which they will operate on any object, and to facilitate ease of switching between levels without causing prejudices to the object itself. That requires a mechanism able to provide different ways of representing a given object (musical or not) – we choose to use the popular "Model-View Controller" pattern (Buschmann *et al.* 1996). The conceptual mechanism for interfacing views on the different objects we are using is outlined in Figure 4.

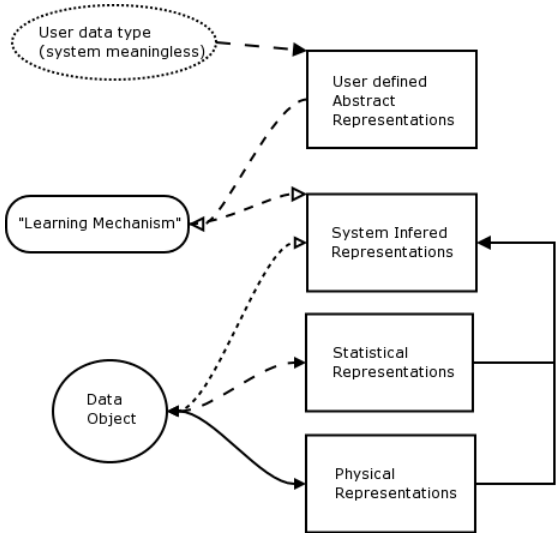

Figure 4: An overview of the abstraction levels mechanism

5.2 Internal Design

From the outset it was clear that the prototype needed strong object orientation in order to be able to implement easily the object model for the basic representation structure. Specifically, we needed to be able to address heterogeneous data types within the context of a single musical artifact. To this end, an abstract class called "SoundStructure" is defined. This class can be considered as an interface masking low-levels calls to I/O mechanisms, and providing a common and consistent way of addressing data structures.

Handling audio and MIDI streams in a consistent way is a difficult problem, which has been partly solved by using a symbolic representation of the audio signal. Specifically, we are investigating the use of synchrony strands (Cooke, 1991) (Godsmark and Brown, 1999), a time-frequency description obtained from the output of a cochlear model. This provides a symbolic representation of significant acoustic components (such as harmonics and formants), which can be grouped using Gestalt principles to derive properties akin to pitch, loudness and duration. Hence, synchrony strands allow audio to be manipulated and abstracted in the same way as a MIDI signal.

Low level representative views are interfaced directly to this basic class, whereas, as described in previous sections, the more complex and higher level representations are handled through completely different objects, issued from classes called AbstractFeature and StatisticalFeature. Most of the complex tasks dealing with interpreting the original data are done from within these classes.

The user interface is constituted of several classes which provide views on the object, as well as a special class to provide support for the user-defined data input (i.e. tablet input), and several classes that let the user access all other classes in the system via a scripting interface.

Another feature is a mechanism which provides an effective way to record composers' behaviour and store it as a database of "gestures" which can be reused in order to constitute a customisable library of composition schemes (the so-called "learning mechanism" in Figures 4 and 5). As well as providing the system with an "understanding" of the relations between a high-level manipulation and the physical counterpart, this will ultimately provide a better insight into the composition process in itself. This mechanism is currently under investigation.

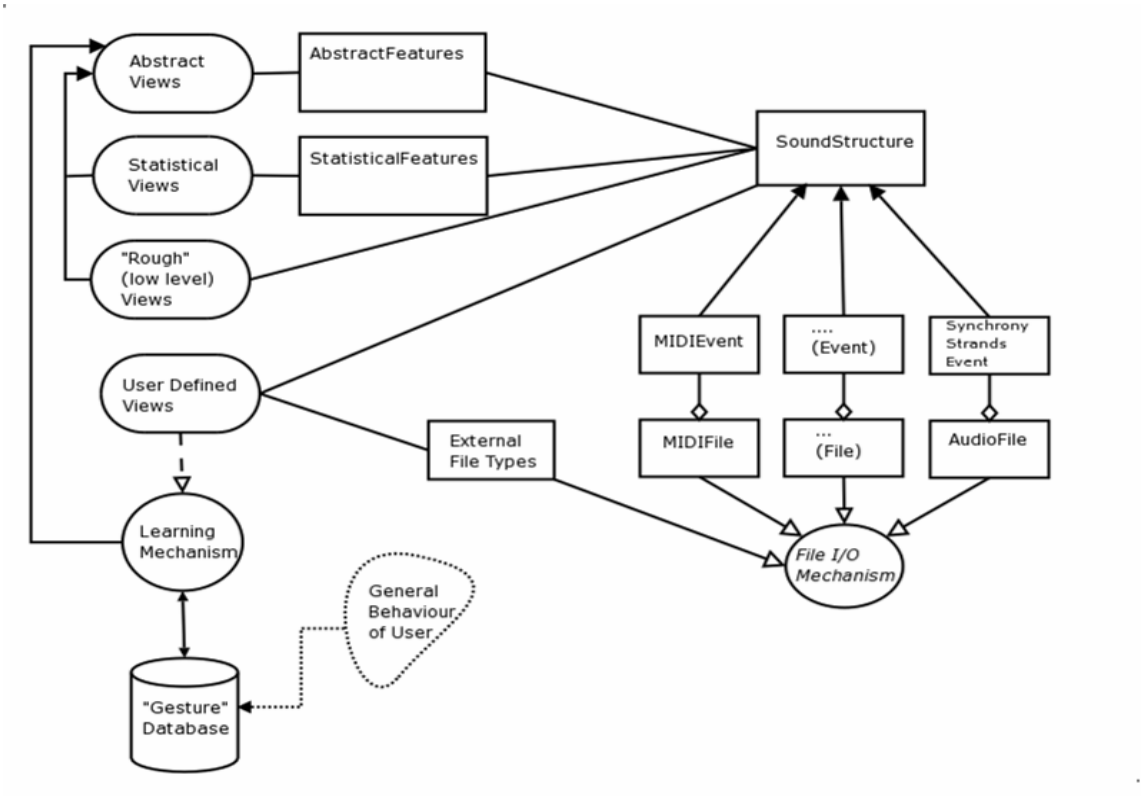

Figure 5: Technical architecture of the prototype

6 Conclusion

In this paper, we have presented a snapshot of research towards composition software that provides new functionalities for electroacoustic music. By strongly basing our analysis of the situation and design on a thorough investigation of composers' habits and behaviour when composing with computer (Nuhn *et al.* 2002), we described the current architecture, both technically and conceptually.

The main problem discussed was to define a coherent strategy to deal with complex musical structures involving multiples levels of representation and relationship. Through the use of object-oriented software design, we have been able to propose a new kind of architecture, based on user-controllable and definable "views" that gives composers full control over every levels of abstraction they chose to use in the composition process. The use of an untyped workspace permits composers to freely associate any object at any abstraction level to another one, without requiring the system to function in a particular "mode" – that is, there is only one level of interaction in which all operations are done. Finally, a learning mechanism is planned and currently under investigation, to permit the system to store statistical data and ultimately create a model of composers at work.

7 Acknowledgments

This work in progress has been funded by the EU MOSART network.

References

Assayag, G., Baboni, J., and Haddad, K.(2001). *OpenMusic 4.0 User's Manual*, IRCAM.

Budon, O. (2000). "Composing with objects, networks, and time scales: an interview with Horacio Vaggione." *Computer Music Journal* 24(3):9-22

Bukhres, O.A., Elmagarmid, A.K. (1996). *Object-Oriented Multidatabase Systems: A Solution for Advanced Applications*. Prentice-Hall.

Burton, A. R., Vladimirova, T. (1999). "Generation of Musical Sequences with Genetic Techniques." *Computer Music Journal* 23(4):59-73.

Buschmann, F. ed. (1996). *Pattern Oriented Software Architecture Vol.1: A system of Patterns*. Wiley & Son, Hoboken.

Cooke, M. (1991). Modelling auditory processing and organisation. Cambridge University Press.

de Bono, E. (1987). Lateral thinking. In: Richard L. Gregory (ed.) *The Oxford companion to the mind.* Oxford: Oxford University Press.

Eaglestone, B.M., Ford, N., Nuhn, R., Moore, A., Brown, G. (2001). Composition systems requirements for creativity: what research methodology? *Proceedings of Mosart Workshop on Current Research Directions in Computer Music*, Barcelona. pp 7-16.

Ellis, D. (1993). Modelling the information seeking patterns of academic researchers: A grounded theory approach. *Library Quarterly*, vol 63, no. 4, pp 469-486.

Guilford, J.P. (1967). *The Nature of Human Intelligence*. McGraw-Hill, New York.

Godsmark, D. and Brown, G.J. (1999) “A blackboard architecture for computational auditory scene analysis.” *Speech Communication* 27: 351-366

Lincoln, Y.S., Guba, E.G. (1985). *Naturalistic inquiry*. Sage Publications; California.

Meyer, B. (1997). *Object Oriented Software Construction*, 2nd ed., Prentice Hall, Upper Saddle River.

Nuhn, R., Eaglestone, B. M., Ford, N., Moore. A. J., Brown, G. (2002). “A qualitative survey of composers at work.” *Proceedings of the International Computer Music Conference*. International Computer Music Association, pp. 572-598.

Puckette, M. (2002). “Using Pd as a score language.” *Proceedings of the International Computer Music Conference*. International Computer Music Association, pp. 184-187.

Reybrouck, M. (1997). Gestalt concepts and music: limitations and possibilities. In M. Leman (ed.) *Music, gestalt and computing.* Lecture notes in artificial intelligence, no.1317. Springer-Verlag, Berlin.

Roads, C. (1998). *La transformation et la synthèse des micro-sons*, PhD thesis, Université de Paris 8.

Vaggione, H. (2001). “Some ontological remarks about music composition process.” *Computer Music Journal* 25(1):54-61.

Windsor, L. W. (1995), *A perceptual approach to the description and analysis of electroacoustic music*, PhD Thesis, Sheffield University.

Yadegari, S. (1991). “Using Self-Similarity for Sound/Music Synthesis” *Proceedings of the International Computer Music Conference*. International Computer Music Association, pp. 423-424.