Musical Information Processing Reflecting its Structure

March, 1999

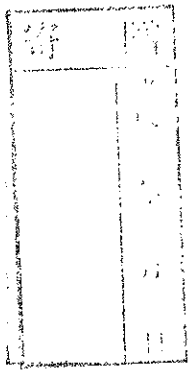
[illegible]

UNIVERSITY OF TSUKUBA

Musical Information Processing Reflecting its Structure

March, 1999

Rumi Hiraga



Abstract

In pursuit of generating expressive musical rendition with rules, the computer music project Psyche has greatly concerned musical structure. Although described implicitly, musical structure exists innately and absolutely in musical scores. This thesis demonstrates the successful introduction of musical structure to computer music systems that are related to performance synthesis. Two systems, a performance visualization system and a computer-assisted musical analysis system Daphne, are described.

A performance visualization system proves the significance of musical structure for analysis and synthesis of performance. Since musical structure and relationships between its occurrences regulate performance expression, the rationale of performance can be explained at a glance with visualized performance data reflecting musical structure. Comparison by the visualization enables users to find resemblances between performances which derive performance rules. The innovative figures of performance visualization is implemented to a system on which users can synthesize performance as well as analyze it.

Based on musicology, the attributive information of musical structure as items of performance analysis are clarified through the design and implementation of a musical analysis system Daphne. Users can obtain the indispensable information for performance synthesis by Daphne through its realistic user interface with score images and performance. Two matters on computer music systems were figured out: the difficulty in evaluating them and the applicability and limitation of object-oriented approach to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis concerns the representation, acquisition, and manipulation of musical structure in generating expressive musical performance by computer systems. Though implicitly described in scores, musical structure exists absolutely and plays an important role in musical rendition.

Music is represented in different abstraction levels of information depending on which musical activities to center. Musical structure belongs to a higher abstraction level where we can find note groups of several kinds to represent musical meanings such as melody, rhythm, harmony, phrasing, and articulation, which are terms used commonly to represent musical activities by human.

Acoustic signal is the most low level information to convey absolute value of music. Sound synthesis as a research area of computer music handles acoustic signal whose purposes are to propose sound components for new types of computer music, to analyze acoustical characteristics of sound, and to build systems for understanding and synthesizing sound [Osa98]. Sound source separation and beat tracking use acoustic signal for analysis purpose. Bitmap is another type of raw musical data which is used by optical music recognition (OMR).

MIDI data are more abstract than acoustic signal because some values of them can be interpreted in more than one way. For example, a value for dynamics represented by hexadecimal may be translated to decibel of sound pressure levels, the ratio between the intensities of the direct and indirect sound field which represents the human hearing system better, or values corresponding to each instrument with MIDI interface. Still MIDI data are

usually not considered abstract because they are independent each other and there are no ways to group them together for representing musical meaning.

A musical score is an established representation especially for classical western music that includes indications on expression and performance as well as information of each note to transmit composers' intentions. Human performers play music by interpreting it to derive the musical meanings for agogics and dynamics based on musical structure.

Some of the higher level musical information is implied in musical scores that is derived with musical knowledge. Research of music cognition pursues to clarify the process of musical understanding through the higher level information. As the research focuses on "human", it cannot neglect non-mathematical elements such as tension, expectancy, and emotion.

There is another approach to clarify human activities on music by constructing systems, which is called *analysis by synthesis*. It builds a model of analyzing musical activities, then the model is verified by synthesizing the musical activities on the systems. Dannenberg described in his paper [Dan93],

> Computers allow (and require) a formal approach to the study of music representation. Computer programs demand that every detail of a representation be precisely specified, and the resulting precision allows experimentation and testing of new representations.

Though the above quote refers only to musical representation on computer systems, the same can be said to acquisition and manipulation of musical information through computer systems.

1.1 Performance by Human Players

Musical activities by human in the real world are described abstractly, conceptually, qualitatively, and often with terms for emotion. Such representation is not understandable to computer music systems. When a famous pianist gave a lesson, she instructed students saying "passionately" or "emotionally" which is understood by "human" students to improve their performance however ambiguous and abstract the instructions were. A musical review for a conductor written as a newspaper article consisted mainly

of conceptual expressions such as "deep besides sincere and steady music." Readers of the newspaper could understand what the reviewer meant by his expression.

A professional player listed some performance plans which he obtained through his musical activities [Fuj95]. Regardless of its applicability to any music, an example is,

R 1 *The middle of a phrase is played faster.*

Though we can understand the above sentence in a sense, the interpretation of the sentence may be different by each person. Even though a phrase were identified, there can be disagreement on the "middle" of the phrase. Moreover, no one can concretely and precisely states the rate and compared portion of "faster".

Musical information we use ordinarily involves natures which must be concerned in building and evaluating musical information processing systems. They are the ambiguity of musical representation by human [Hir98b] and the indefiniteness of the whole set of musical knowledge [Bal96][Dan93][Pop91a].

The above plan shows the ambiguity of representation. Indefiniteness for performance rule set means that how huge the number of performance rules becomes, it can never be confirmed that the rule set is complete enough for generating expressive performance.

1.2 Performance Synthesis on Computer Music Systems

Performance software has the long history in computer music research: from punched paper-tape in 1950s, analog and digital sequencers in '70s, to computer controlled instruments these days. Since MIDI protocol was introduced in 1983 into the market, research on performance synthesis could concentrate more on musical contents.

In order to give performance synthesis systems the value of artificial intelligence which instantiates human activities, the following are necessary stages: the theoretical research of formalization of musical information and the implementation of systems based on the theory. Between the two stages,

there lies designing a computer language for music, quantification of abstract representation, and visualization.

As can be seen in the example performance plan R1 in the previous section, empirical performance rules, as formalized representation of performance plans, are usually described as follows.

1. The premiss of a rule specifies a portion of a score to which the rule is applied.

2. The conclusive of a rule is qualitatively described how performance parameters should be.

In order to generate performance using these rules, two issues should be solved.

1. To specify where to apply a rule.

2. To quantify performance parameters expressed qualitatively in a rule.

There can be some transformations of the above plan R1 into the more concrete representation for a computer system to understand. If a transformation were "when a note is in the middle of a phrase, then the span between two onset time of the middle note and the next is specified shorter", a system can understand this transformation because the original term on tempo ("faster") is changed to the output-oriented representation, assuming the output is in the MIDI format, although still qualitatively described ("shorter"). When the "middle" is interpreted in a way, such as the middle point regarding the beat count in the portion, the applied portion of the rule can be calculated with the score information and the specification of "phrase". If the "middle" is at the metrical accent, the applied portion can be found by musical analysis.

1.3 Musical Structure for Performance Synthesis

The "phrase" in the plan R1 is an occurrence of musical structure, which is a key to solve issues described in the previous section. The empirically obtained performance plans are reflected by musical structure, since human players

take musical structure into account when they represents their interpretation of music as performance. Although musical structure is not explicitly denoted as symbols on a score, its definite existence has been known to professionals of music.

Computer music researches which are closely related to performance including automated accompaniment systems utilize information of musical score and the higher abstraction. There is the "egg and chicken" relationship between performance and the information of the higher abstraction level. With a score as the source of basic information, performance is synthesized using the higher abstraction level; on the other hand, the higher abstract information can be obtained from performance data. In this way, performance and information of the higher abstraction level, especially musical structure, are indispensable to each other.

Musical structure, as information of the higher abstraction level, can be obtained either manually or automatically through a system. In any ways, musical structure obtained through a system has no ambiguity on representation. Thus the intention of a human reflected to the analyzed structure is transferred to a system through which the structure can be shared among people and among systems.

This research is a part of a computer music project Psyche (Program SYstem-Conducted Harmony and Expression) which have been pursuing to generate expressive musical performance of classical western music. Psyche's performance generation systems use performance rules confirmed by performance analysis among candidate plans derived from musical common sense and experiences. These empirical rules are originally described in the same way as the one in Section 1.1 and other musical activities: conceptually, objectively, qualitatively, and with terms for emotion. As shown in the previous section, these plans are transformed to another representation with musical structure for computer music systems to understand.

1.4 Organization of the Thesis

This thesis proves the importance of musical structure in analysis and synthesis of performance by building systems.

An integrated performance visualization system based on musical structure was implemented. The rationale of performance can be explained at a

glance with a newly developed figures. Performance is generated and modified with functions reflecting musical knowledge on the system.

A musical analysis system Daphne (Declarative Analysis of PHrasing aNd Expression) was designed and implemented. Through the design of Daphne, we have clarified the musical knowledge necessary for performance synthesis among which musical structure is the most important. In order to share and discuss about the knowledge of musical analysis with professionals in music, score images and performance are involved in the user interface so as to make it imitate the real world analysis. We have discussed issues on evaluating computer music systems and shown by implementation the applicability and limitation of the object-oriented approach for representing musical knowledge for performance.

Chapter 2 describes the role of musical structure for performance synthesis with theories and systems. Musical knowledge to be analyzed and used for synthesizing performance is also shown. In Chapter 3, we will briefly describe the computer music project Psyche: the research strategy, equipment, and systems. Chapter 4 and Chapter 5 will deal with systems concerning musical structure. In Chapter 4, an integrated performance visualization system is described on which performance data are visualized for analysis and edited for synthesis. In Chapter 5, a computer-assisted musical analysis system Daphne is described with discussion of evaluation method for computer music systems and utilization of existing software methods for them. We will conclude this thesis in Chapter 6. An application of attribute grammar to performance synthesis is introduced in Appendix.

Chapter 2

Musical Structure and Performance Synthesis

Computer music research, especially related to automated performance, was rather *score-oriented* in a decade ago [NS87], while the approach is more *structure-oriented* in these days [NHHH98].

In the era of score-oriented, computer music systems for synthesizing performance data utilize information explicitly shown on musical score mainly as their input. There are a few ways to obtain information described on musical scores: optical recognition, acoustic data of performance, and languages to represent symbols on a score [Tag87]. These methods for obtaining score information were research topics in those days.

Psychology and cognitive research on music have noticed the importance of musical structure [CH87][Deu82][Hat87] [HCW85][HWC91]. On the other hand, it is not so long when the importance of musical structure to computer music systems is noticed and utilized in them. Systems which related to performance have greatly progressed once concerning musical structure.

Musical structure is used commonly among several musical activities, whereas it is implicitly described on a score. Composers embody their musical intentions in musical structure in their works. A theme appears in many places in various styles in a piece to give it the completeness and consistency according to Reti's analysis on motif [Ret95] (though it has the critiques of extremeness [Ock91]).

Players in distance in time and space can render the works with information of musical structure as well as explicitly notated information on a score.

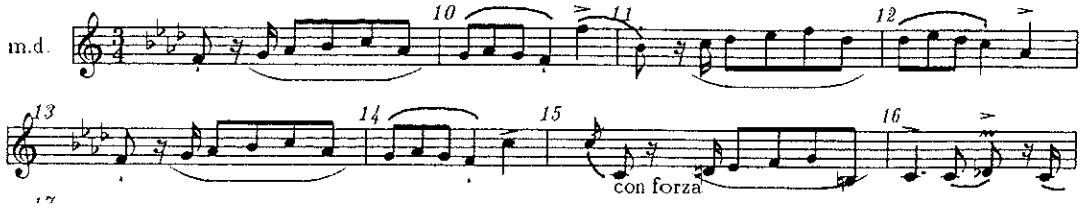


Figure 2.1: Mazurka Op. 7, No. 3, by Chopin (right hand) —MusicTEX

In his article [Slo82], Sloboda explains the role of structure to performance as follows.

> A performance plan that is adequate as a basis for effective expressive variation must therefore contain information about the rhythmic and tonal structure of the music. In such a plan individual notes cannot be thought of as simply separate items standing in some fixed relationship to other individual notes. Rather they must be thought of as elements within superordinate structures whose nature determines the functions of the items. It is hard to see how a musician could perform expressively, except by rote-learning a set of expressive "tricks" for each new piece, unless his or her knowledge of the music were couched in terms of these structures. Sight-reading an unedited score expressively would certainly be impossible.

Thus musical structure is a medium between composers and players (though one way communication) which is used in rendering music as follows.

- First of all, notes on a score are never independent each other. Some may have closer relation with the neighboring notes and some conspicuous than neighbors. The relation and the role of each note are clarified because we take musical structure in concern.

  In the sample score in Figure 2.1, starting from the ninth measure, performers play the eighth notes in the ninth and tenth measures in legato with the slur symbols. The slur is a way to indicate the existence of musical structure. Though not slurred, the ninth and the tenth measures are played in a move because they are in a larger structure.

The f^2 of the third beat in the tenth measure is played with accent not only because there is an accent symbol but also it is the center of the larger structure from the measure nine to twelve.

- Expression curve appears uniquely to musical structure.

  As for dynamics in the first four measures, the curve is in the shape of height ($\frown$), never dale ($\smile$), decreasing ($\searrow$), nor increasing ($\nearrow$).

- Using melody, rhythm, and other grouping, musical structure is derived from score syntactically. The sample score can be divided to the structure as in Figure 4.6 (page 49, the sample score corresponds to a sentence labeled stc1). Thus we can see the sample score's constitution as "$AA'AB$."

  The syntactical similarity between the first and the third structure should be concerned in rendition, namely to give them the semantical similarity. Such a relationship derives the performance for the two structures as

  - being resemblance, but
  - not the same.

  The resemblance is derived because of the syntactical similarity. The second "not the same" means that players intend to play differently the second "A", because it is the third occurrence in the sentence.

  Performance should be constituted not only by concatenating that for each occurrence but also concerning the position of the occurrence in the larger structure.

Musical structure is, intuitionally, the division of a piece. It also refers to group of consecutive and concurrently appearing notes on a score. The group is constituted when it has musical meaning in some sense, such as a motivation of the music (motif). We call *occurrences* of musical structure as concrete group of notes on a score.

In this chapter, the nature and researches of musical structure are introduced. Then compared are some approaches of computer systems for performance. Finally, how our project, Psyche, makes use of musical structure is briefly explained and the structure of musical knowledge for performance is introduced.

2.1 Theories on Musical Structure

The musicological research on structure started in the nineteenth century by H. Riemann whose theory on musical structure [Rie84] and harmony have affected research of musical analysis in these days. His theory on structure takes eight measures as a norm in which two and four divisions also have musical meanings. Based on the phrasing analysis by Riemann, we call two measures *motif*, four measures *phrase*, and eight measures *sentence*. His system can analyze especially the roman style well [Kel87]. Famous composers such as Bruckner and Schoenberg succeeded his theory of the importance of the eight-measure norm, in spite of several objections to his theory, for example how to explain Beethoven's frequent third measures in motifs or neglect the actual size (reflecting performance) of each measure.

Starting from Schenker's Analyse, some theories of psychological and cognitive analysis of music have been proposed [Mur87]. Schenker's theory introduced hierarchical structure depending on the theory of harmony. By prolonging significant notes, musical works are divided to three types of Ursats (fundamental structure). The drawback of this theory is the neglect of other musical elements —rhythm, melody, dynamics, and agogics— thus it could not explain how listeners understand structure by the reciprocal relation of these elements.

Meyer and Cooper introduced Gestalt psychology to Schenker's theory to constitute group of notes by modeling the *Implication-Realization* (IR) model [Mey56] and rhythmic structure [CM60]. IR model is refined with Narmour to explain music by the relationships of consecutive notes: the previous note lets listeners expect the following ones and the following explains or confirms the previous.

Lerdahl and Jackendoff proposed *Generative Theory of Tonal Music* (GTTM) in their book [LJ83]. This theory tries to clarify cognitively the understanding process of music by musically well-trained people. Based on generative grammar by Chomsky, GTTM proposes four groups.

- Grouping structure is the hierarchical segmentation of motifs, phrases, and sections[1].

[1] "sections" correspond to "sentences" in Psyche. It is often the case the name for structure is used differently (one of the ambiguity of musical representation). For example, Schoenberg usually uses "motif" for a measure.

- Metrical structure is the regular and periodical change of strong and weak beat. This structure also forms the hierarchy.

- Time span reduction assigns the structural importance to each note in the grouping and metrical structure.

- Prolongation reduction builds a hierarchical structure where harmonic and melodic tension, relaxation, continuity and progression are found.

These theories can be computational models, among which GTTM is the most widely adopted by computer music systems [Deg96][UKI97][Wid93]. We have to be careful though, the value of a theory in applying to a system we try to build, however impressive a theory itself. Balaban claims that a music system cannot be built based on any one theory [Bal96].

For example, constraint view and insufficient explanation lead inconvenience in building a system of performance synthesis.

- Constraint occurring from the theory to explain cognitively the music understanding.

  - Grouping structure in GTTM is successive. Although it may be able to explain the understanding process, consecutive structure in all levels is not necessary for performance synthesis.
  - Priority is not given to constituents, while performance is built in a sense selective, since there are the more important and the less important groups. This selection is not involved in GTTM.
  - Metrical structure in a level has the same length. With the same reason as the grouping structure (selection is not involved), GTTM's metrical structure is not suitable to automated performance synthesis.

- Insufficient explanation

  Like Reti's detailed structure relationships were exemplified only with artistic explanation, some theories describe the expectation without explanation.

Possibilities and limitations of each theory in applying to computer music systems are described by Katayose [KT94]. We conclude that theories on

musical analysis are in a mess and using any one of these theories is not suitable to the actual systems of performance synthesis. Thus musical structure in our research is based on musicology which is generally accepted.

2.2 Performance Synthesis Systems

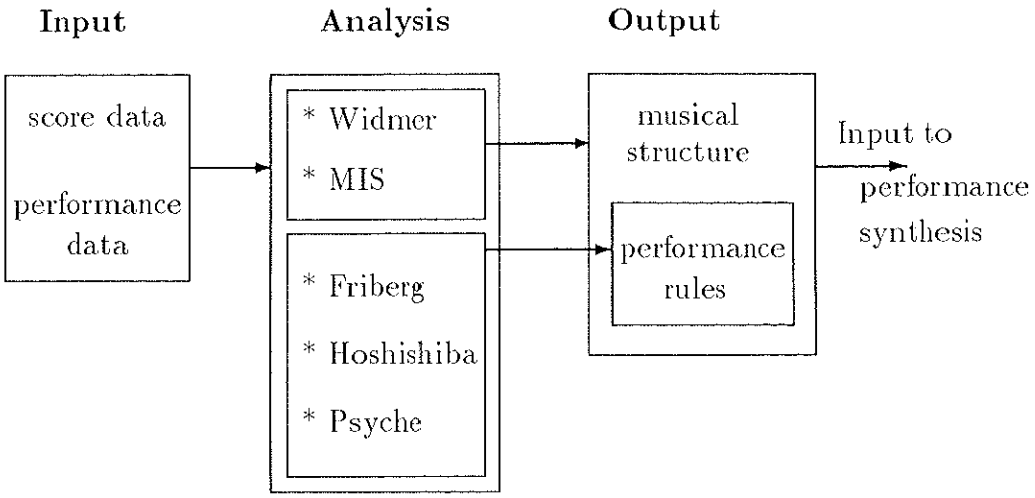
Performance synthesis is a process of a computer music system in order to obtain musical rendition as its output. As described earlier in this chapter, current synthesis is structure-oriented. Namely, we have not contented with the output of systems using only the information explicitly appearing on a score but also require other higher level information to improve the output.

Performance analysis as a process of performance synthesis

Another difference in today's performance synthesis systems from those in a decade ago is that there is a process of performance analysis before the synthesis [Hir98a]. Analysis uses both score data and performance data of either MIDI or acoustic signal. The purpose of the analysis is to derive musical structure and/or performance rules. Some systems analyze performance to obtain both information (musical structure and performance rules). These are MIS (Music Information System) by Inokuchi lab., Osaka University [AKI97][FKI90][KFI90] and the famous Widmer's system based on GTTM [Wid93][Wid94].

Some obtain one of the two (usually performance rules). These are Friberg and Sundberg's system [Fri91], Hoshishiba's statistics based system [HHF96], and Psyche's system with empirical performance rules [IHI97][II95][KIIH98].

The former systems automatically gain musical structure because they are based on a cognitive model where deriving structure is an issue. The latter systems don't make much concern the automation of obtaining musical structure, however they use musical structure at the performance synthesis stage. They are the more synthesis-oriented systems. Both types of systems use musical structure and performance rules as input to performance synthesis. Figure 2.2 summerizes the performance analysis as a preprocess of performance synthesis.



Widmer's and MIS automate the aquisiton of musical structure because they are based on cognitive models where deriving sructure is an issue.

For systems by Friberg, Hoshishiba, and Psyche, automatic aquisition of musical structure is not a big point.

Figure 2.2: Performance analysis as a process of performance synthesis

Performance synthesis with musical structure

There are four types in synthesizing performance data: *performance rules*, *case-based reasoning*, *neural network*, and *graphical user interface* (GUI). Among these, systems to use performance rules and GUI have noticed the importance of musical structure for synthesizing performance.

- Honing proposed a performance editor to synthesize musical performance. In his paper [Hon92], he claims that it is not the standardized performance but an interpretation of an important occurrence of musical structure that regulates the expression.

- Though not in the stage of performance synthesis yet, Chafe compared two musical renditions using two variables of onset time and dynamics of each note [Cha97]. He claimed the importance of comparing renditions of occurrences of musical structure. Comparison that is not structure base is meaningless for obtaining information on performance expression.

- Widmer built a system for performance synthesis in 1993 [Wid93] based on GTTM. In the performance analysis stage, the system automatically obtained musical structure (grouping structure and prolongation reduction) from performance data and score data. At the same time, performance rules on a note were also derived. Then at the performance synthesis stage, the system generated musical rendition for another music. The conclusion was not so satisfactory and the reason was analyzed by the author as the rule is on each note, not on musical structure. In the next system [Wid94], rules were on musical structure.

- MIS was developed by Inokuchi lab., Osaka University. Performance rules were obtained with multiple regression analysis based on musical structure and a score.

- Psyche's performance synthesis systems obtain rules by analyzing performance data. Rules are applied to occurrences of musical structure.

Table 2.1 summerizes the systems related to performance synthesis which take musical structure much in concern. Table 2.2 summerizes systems not shown in Table 2.1, which devote to performance synthesis. Papers by Suzuki [STT97], Murakami [MKKN96], and Bresin [BDPV92] describe their systems respectively.

2.3 Psyche's Approach

Musical structure, including the relationships between occurrences, has been the subject of the musical analysis research. Although the importance of musical structure has also been admitted by creative musical activities including performance synthesis, the result of analysis is not necessarily the premiss for synthesis.

Author	System	Purpose
Chafe (CCRMA, Stanford Univ.)	comparing performance	performance analysis
Honing (Univ. of Amsterdam)	a performance editor with GUI	performance synthesis
Widmer (Univ. of Wien)	GTTM based performance analysis and synthesis	
MIS (Inokuchi lab., Osaka Univ.)	extract parameter values with multiple regression analysis	performance synthesis
Psyche (Igarashi lab., Univ. of Tsukuba)	derive and apply performance rules	performance synthesis

Table 2.1: Systems which take musical structure in concern

Author	Synthesis type	Input
Friberg (Royal Institute of Technology, Sweden)	rule	score, musical structure
Hoshishiba (JAIST)	rule	performance, score
Suzuki (TIT)	case-based reasoning	score, musical structure
Murakami (Kansai Univ.)	neural network	performance, score
Bresin (Padua Univ., Italy)	neural network, rule	score

Table 2.2: Performance synthesis systems

Those systems which made success in computer music have noticed the significance of musical structure to the system. They made success because musical structure can explain the rationale of performance. In his paper [Hon92], Honing described the performance expression as follows.

> We proposed to define expression in terms of a performance and its structural description. We can then define the expression of a structural unit as the deviations of its parts with respect to the norm set by the unit itself.

Though it represents the effect of musical structure to performance, the "deviation" is still abstract to be applied to building systems.

Based on our yearly experiences, we have known the concrete effects of musical structure to performance in several ways as described in pages 8 through 9 and built performance synthesis systems by including information on musical structure which must be obtained through a system also.

The project Psyche obtains analysis information through a musical analysis system Daphne (described in Chapter 5) then uses it for performance synthesis. In other words, the project bridges two types of musical activities —analysis work and creative work— to form an overall musical activity.

In the following subsection, we will describe concretely the structure of musical knowledge consisting of attributive but necessary information.

2.3.1 The structure of musical knowledge

Musical analysis is the embodiment of musical knowledge on a piece. Since analyzed information on the system Daphne is related to performance, it can represent composers' intentions in performance. In analyzed information, there is no ambiguity which exists in human conversation described in Section 1.1 (page 2), therefore users of Daphne can share an interpretation of the piece. The information stored in Daphne can be shared among systems of Psyche also.

Figure 2.3 shows the structure of musical knowledge obtained through analysis. In this figure, we can observe two innate characteristics in musical structure.

1. The hierarchical relationships.

2. The contextual relationships regarding time.

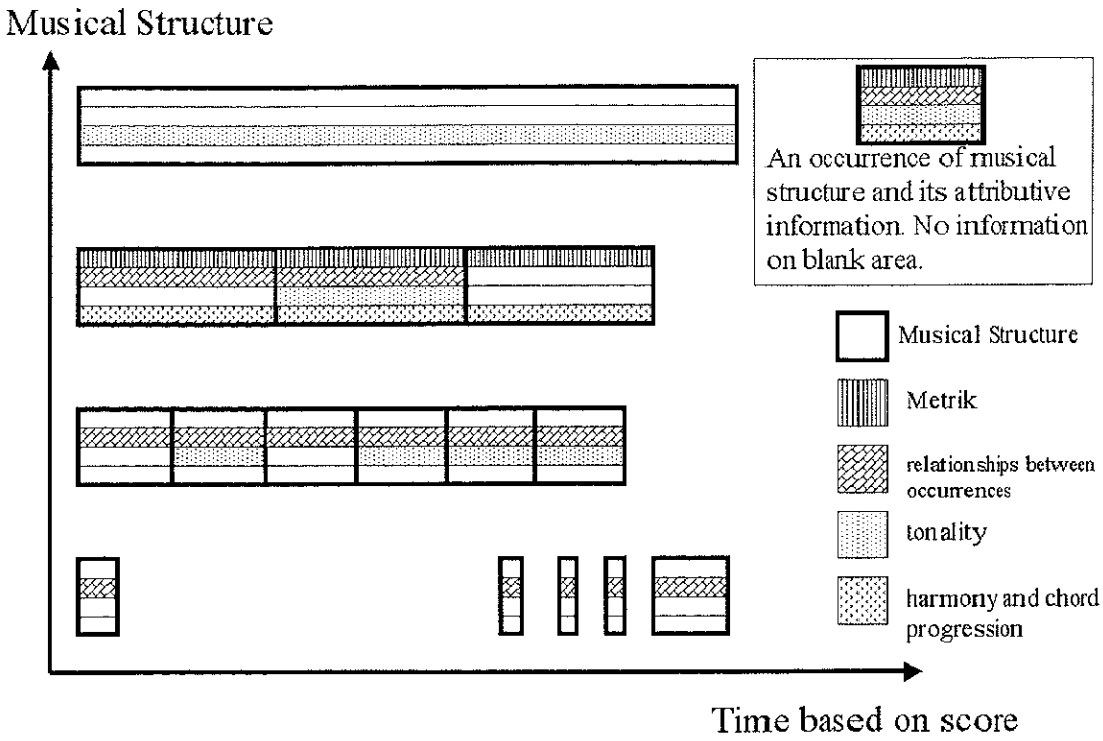


Figure 2.3: The structure of musical knowledge

As for the contents of occurrences of musical structure, two characteristics can be observed.

1. Some occurrences have relationships with others.
2. Each occurrence involves other attributes in music, such as metrical structure and tonality structure.

Though only three units of musical structure, motif, phrase, and sentence, are mentioned in Chapter 2, there can be any levels of musical structure in pieces actually. The figure shows the unrestrictedness in introducing any level of musical structure, for example *submotif* which is smaller than motif, and the spontaneous analysis on the structure. The spontaneous analysis

means that the whole piece is not necessarily analyzed into occurrences of musical structure of all levels. A user can specify the necessary and enough structure for performance synthesis.

Other four items of musical analysis are found based on each occurrence of musical structure. In other words, they are attributes of musical structure. It is sometimes the case that other items of analysis specify an occurrence of musical structure, however analysis is confirmed after occurrences of musical structure are specified.

Because of the indefiniteness of musical knowledge described in Section 1.1, computer music systems should be designed concerning extensibility for increasing the type of knowledge. A musical analysis system must have a mechanism to extend items to be specified for increasing number of rules in the future. A new item of analysis introduced in Daphne will be handled as the same way as other structure-attributive items.

In Figure 2.3, two types of inheritance based on hierarchical structure are shown.

1. Performance characteristics of the upper hierarchy are inherited to the lower ones.

   An example of this performance inheritance will be described in 4.2.2 *Examples* (page 42) as a reason of performances of two resembled occurrences are partially different.

2. As musical structure constitutes a hierarchy, so as other analysis items do (*recursive hierarchy*).

Explanation of the structure with Beethoven's Sonate

Musical structure in Figure 2.3 is described with the first sixteen measures of the first movement of Beethoven's sonata, Op. 57 (*Appassionata*), which is one of the greatest, noteworthy, and most famous among Beethoven's sonatas consisting of three movements. The beginning of the movement is shown in Figure 2.4 indicated with the elements in the commentary[2].

The following commentary is by Moroi [Mor].

[2]The score is drawn with Kawai's ScoreMaker [Kaw98] from the scanned original score [Mor]. Because of the restriction in the software, we could not completely reconstruct the original score.

Two elements in contrast are both in the exposition of the first movement. The elements are unfolded to constitute all movements.

The first movement, Allegro assai, F minor, 12/8 time, is in the sonata form[3]. The first exposition consists of sixty-five measures where the first sixteen measures in the figure is the most significant and has four elements of A through D in the figure.

Numbered lines in the first four measures include two motifs. The first motif (indicated by a line labeled 1) includes a typical melody indicated as A (R1) of *arpeggio*[4]. The first motif itself is constructed with A and its development. The second motif (line 2) appears as harmony where all notes progress in supportive move.

The next phrase, measures from five to eight, has the same construct as the first except the change in the key from F (minor) to Ges (major).

The third phrase, measures from nine to thirteen, is mainly the development of the element B, including the type C whose rhythm named R2 is important. The final, measures from fourteen to sixteen, closes the theme. The fast passage D is an arpeggio which we can regard it to be made as the unfoldment of A. This rhythm (R3) of the sixteenth notes as well as R2 constitute the dynamic move of the piece.

Namely, two generative motifs, A and B, appear in the first phrase from which two derivative motifs, C and D, are generated. Three types of rhythm, R1, R2, and R3, are included here, where R2 and R3 are unfolded to the whole piece.

---

[3]from [Ken98]: regular sonata form implies three sections: 1. Exposition, containing first subject in tonic key, and second subject, in dominant, often repeated and followed by 2. Development, in which the material of the Exposition is worked out in a kind of free fantasia and 3. Recapitulation, in which the Exposition is repeated, though often with modification, and with the second subject now in the tonic. The basis of sonata form is key relationship.

[4]from [Ken98]: a chord "spread", i.e. the notes heard one after the other from the bottom upwards, or sometimes from the top downwards, as on the harp.

Occurrences in Figure 2.5	Measures	Musical Analysis in the commentary
4	1-16	F minor
3-1	1-4	phrase
3-2	5-8	the same construct as 3-1, key from F to Ges
3-3	9-13	
2-1	1-2	motif 1
2-2	3-4	motif 2 (B), key to C
2-3	5-6	resembles 2-1
2-4	7-8	resembles 2-2, key to Des
2-5	9-10	resembles 2-2, key to C
2-6	11-12	resembles 2-2, key to F
A		downward arpeggio
C		developed from B
C2		repeat C
C3		repeat C
D		developed from A, arpeggio

Table 2.3: Structure of Appassionata

Using Figure 2.4 and Figure 2.5, we will describe how musical interpretation is represented to structural information.

An occurrence labeled 4 in Figure 2.5 represents sixteen measures of the score in Figure 2.4 whose tonality is considered F minor with the four flat symbols.

Three occurrences in the next layer are for phrase. Though not explained in the previous commentary, metrical accents and chord progression are analyzed in each occurrence. The second occurrence resembles to the first with the change in the key, from F to Ges, as described in the commentary. The analysis (Metrik, chord progression, and relationships) is shown in Figure 2.5 by corresponding patterns.

The next layer represents motif. Each motif consists of two measures (precisely 2-1 through 2-4 are with Auftakt) and appears consecutively. Occurrences 2-1 and 2-2 represents motif 1 and motif 2 in the score in Figure 2.4. Relationships are found between 2-1 and 2-3, also 2-2, 2-4, 2-5, and 2-6. The key is shifted to C major in 2-2 and 2-5, to Des major in 2-4, and to F minor

in 2-6.

Four elements A through D are in the next layer (B is almost the same as the motif 2-2). C, which is derived from B, is repeated twice in motif 2-6. The arpeggio passage D is derived from A.

Table 2.3 shows the correspondence between Figure 2.4 and Figure 2.5.

In this commentary, a highly knowledgeable interpretation is involved: the development from B to C. Although these two elements are different in rhythm, melody, and pitch, the relationship is found by human musicians and used for performance. It will be a future consideration to take the relationships of this type of development into account.

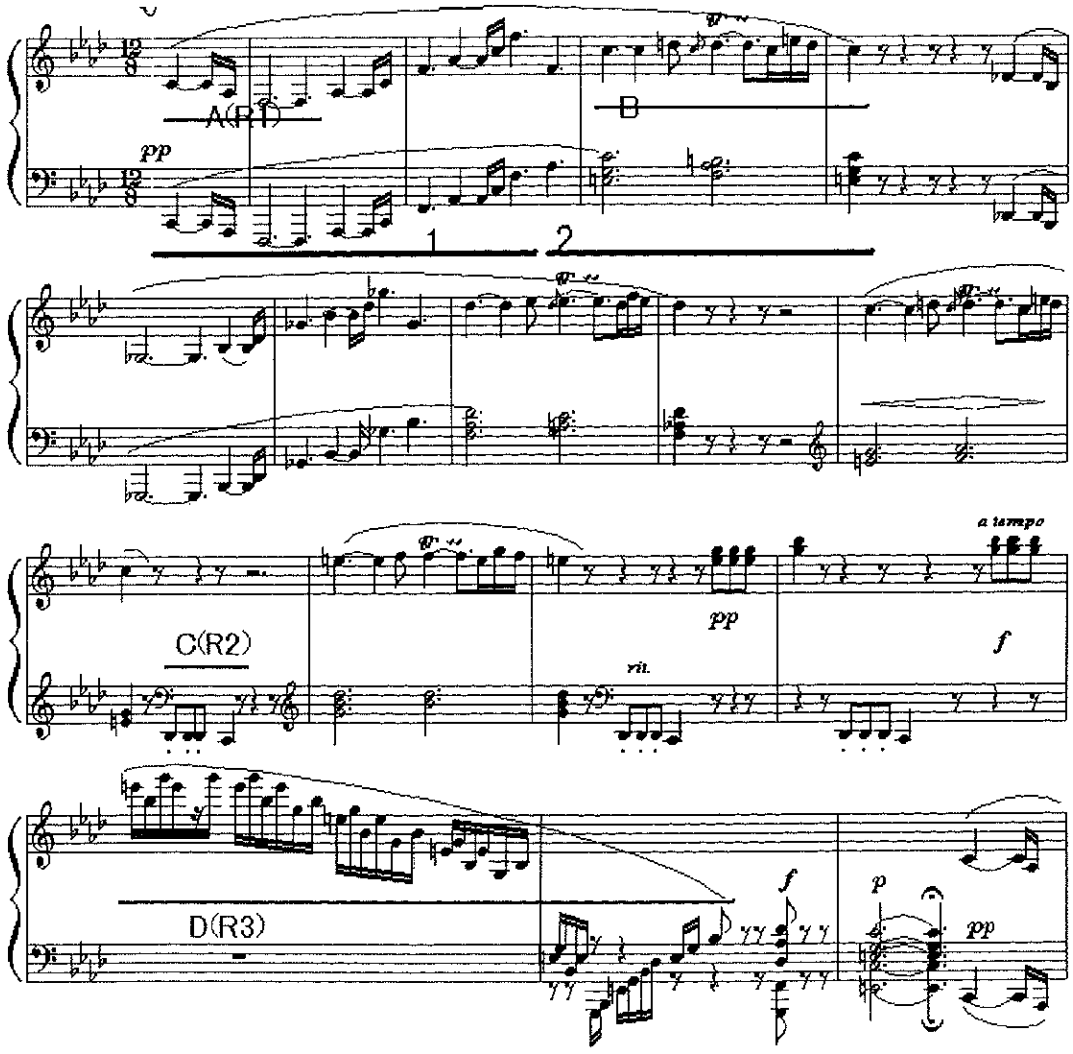


Figure 2.4: Piano Sonate Op. 57, F minor, by Beethoven with the commentary —ScoreMaker

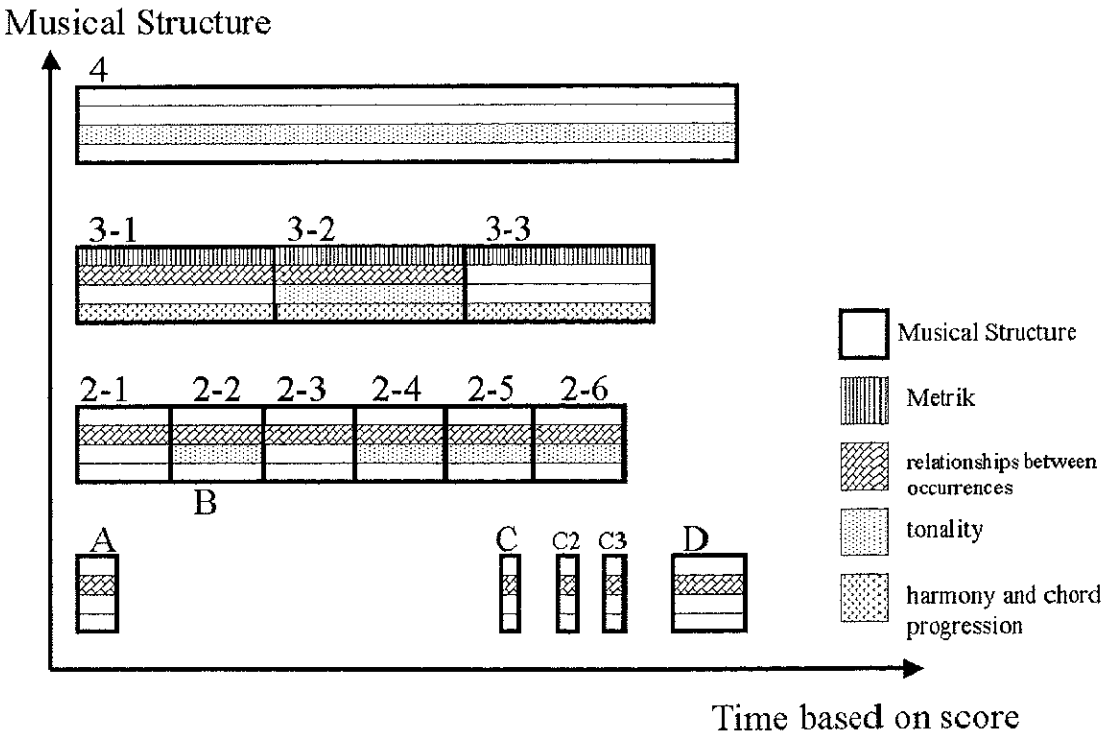


Figure 2.5: Musical knowledge for *Appassionata*

Chapter 3

Computer Music Project Psyche

The computer music project Psyche started in 1983 by Professor Igarashi's group in University of Tsukuba [HI97]. The author was one of the earliest member. The project has been pursuing to let a computer render musically expressive performance of classical music pieces composed mainly in the eighteenth and nineteenth centuries, which are tonal music, consist of structure, and are still widely chosen to be played by professional pianists in these days on an acoustic instrument. In order to pursue the objective, we have built and been building some infrastructures as well as individual computer music systems. The systems are categorized into three types: performance synthesis, automated accompaniment, and performance visualization.

Although the computer music has not a short history and some Japanese groups had been engrossed in the subject in early 1980s, the computer music as a research area has not been pervasive and acknowledged in Japan then. We presented at a symposium our score description compiler language Europa (Extensible Universal Representation Of Phrasing and Articulation) [SMI84], which was used for notating information explicitly appeared on a score. The presentation inspired to formulate an informal computer music research group, now grown to an approved SIG (special interest group) of IPSJ (Information Processing Society of Japan), SIGMUS, which plays the central role to activate computer music research in Japan.

3.1 Research Strategy

In order to obtain musically expressive performance by a computer, many systems have been designed and built in Psyche along the following three directions.

1. *Prepare score data in Europa beforehand.*

   Score data conveys composers' musical intentions beyond space and time. As Oppenheim described in his paper [Opp92], not all of composers' intentions are explicitly denoted on musical score, however we can interpret a score and derive important information for generating performance. In a score, well-trained human performers can find musical structure, relationships between occurrences of musical structure, and harmony and chord progression, for example.

   Europa is a compiler language compiled to generate MIDI code as object. Namely, we can make instruments play notes as precise as they are written on a score by writing a score in Europa.

   The MIDI code was extended for manipulating performance data to introduce a new format called UNI. Because it was not easy for human to read MIDI data which is in hexadecimal code and represents MIDI events, a human readable UNI represents note events along with labels for musical analysis and other information for performance and accompaniment. For example, if a note is resulted as a member of a motif and phrase from analysis, then the note event has labels for the motif and phrase.

2. *Concentrate on the understanding and generation of musical expression.*

   Research issues which are more immediate to synthesize and analyze music rendition are given the priority. Consequently, some research issues of a few steps away from generating performance are not the present objective, however important and interesting they might be.

   For example, analysis of acoustic signal is not our issue to be solved. Thus we manipulate MIDI that has fewer performance parameters than data from other devices, instead of acoustic signal which we have to start from signal processing. Even the higher level information so

closely related to performance is obtained in a "computer-assisted" way. The complete automation of analyzing musical structure is not necessarily a primary issue in Psyche. Users of Psyche systems can either specify or inquire musical structure through a musical analysis system Daphne (Chapter 5).

3. *Introduce musical structure.*

   Systems of Psyche are designed concerning musical structure as will be described in Section 3.3. This turned to be the most effective direction in solving various fundamental problems not only in performance synthesis but also in automated ensemble and performance visualization.

3.2 Equipment

The acoustic grand piano with MIDI I/O and synthesizers are our oldest instruments for experiments. Other instruments with MIDI I/O are accordions, a silent drum system, a guitar, and a clarinet.

The output of systems for synthesizing music rendition is played on the grand piano. There are some combinations of instruments played by accompaniment systems: an ensemble system plays the second in a piano duo, accompanying the primo by a human player, another plays the piano, accompanying melody or beats of synthesized sound by a human player, and the other plays a MIDI drum set accompanying the piano by a human player. Orchestrated data for piano concertos in the market are partly used for an accompaniment system also.

A big issue in using the piano as the output of an accompaniment system is that a system has to send MIDI data of the accompaniment part 500 millisecond prior to the actual piano key movement. 500 millisecond is a restriction to the calculation of the expected time when the next MIDI event to happen.

Accompaniment systems to control MIDI instruments have been built on DOS. Though the inevitable time latency is pointed out informally by a developer of Windows 95, we have to implement a MIDI I/O program on Windows 95 and its successor Windows 98 or even Windows NT considering other factors such as programming environment. In order to make an accompaniment system satisfiably work where the real time control is essential, we

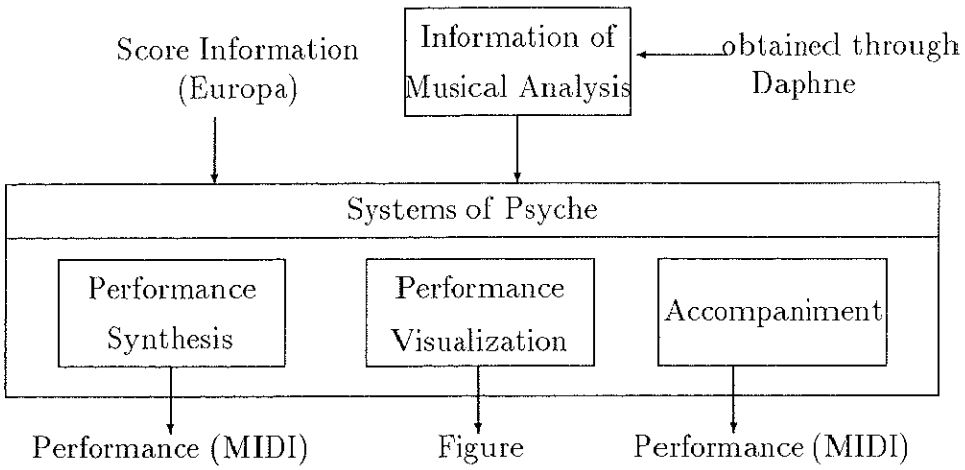


Figure 3.1: Systems of Psyche

have to consider the least time latency caused by an operating system as well as an algorithm for a system to generate musically contented accompaniment.

Static programs, such as synthesizing music rendition and visualizing musical performance, are written both on Unix and Windows 95/98 currently.

We use Visual Basic for graphical user interface (GUI) of a system and Visual C++ for data manipulation if a system is implemented on Windows 95/98.

3.3 Systems

Systems of Psyche are categorized into three: one consists of static programs to synthesize expressive performance with rules, another dynamic real-time programs to control a computer so as to accompany a human player, and the other visualization programs to analyze performance. They are all pursuing expressive performance. These systems require the information of score and musical analysis as input and obtain each output (Figure 3.1). Information of musical analysis is obtained through Daphne, Declarative Analysis of PHrasing aNd Expression, described in Chapter 5. Daphne enables systems to share musical knowledge, thus when new knowledge is introduced to a system for improving musical quality, other systems are expected to be improved as well by the sharing.

3.3.1 Performance synthesis with rules

There are two types of performance rules based on musical structure: one is on relationships between occurrences of musical structure, the other on expression inside an occurrence of musical structure.

An example rule of the first type is as follows.

R 2 *when a consecutive occurrence of musical structure is considered almost the same as the previously appearing occurrence (a seed occurrence), then the latter occurrence (a referring occurrence) is played softer than the first one.*

Since there is a close relationship between musical structure and rendition, rendition is obtained by unfolding performance of some occurrences of musical structure, namely *seed occurrences*, which are usually motif [IHI97][II95]. The performance is expanded by applying *context-free* performance rules like R2 based on musical structure. Context-free rules describe relationships between occurrences of musical structure and appearance of significant harmony progression. Currently, a portion of the actual performance by a professional pianist is used as a *seed performance* for the expansion.

An example of applying the first type rules is shown with the eight measures (a *sentence*) from Mazurka Op. 7, No. 3 by F. Chopin (Figure 3.2). This eight-measure sentence is analyzed to four occurrences of motif which are indicated by four lines under the score in the figure. Capital letters on the lines ($AA'AB$) show the relationships between the four. Make the first occurrence as a norm, the second resembles to the first, the third almost the same, and the fourth can be regarded as an independent motif. In this case, the first and the fourth become "seed" whose expression is unfolded to generate the whole performance of the sample score.

Another type of performance rules prescribes the expression of each occurrence of musical structure [KIIH98]. Performance data by several human players are compared and analyzed to derive the common performance way which are candidates of performance rules. Then by performance visualization as described in Chapter 4, some of them are considered performance rules. They are confirmed to be rules by synthesizing performance.

We can automatically synthesize performance for the whole score by using the latter type to generate performance for seed occurrences, then to unfold the expression by the former type to generate the rest of the music.

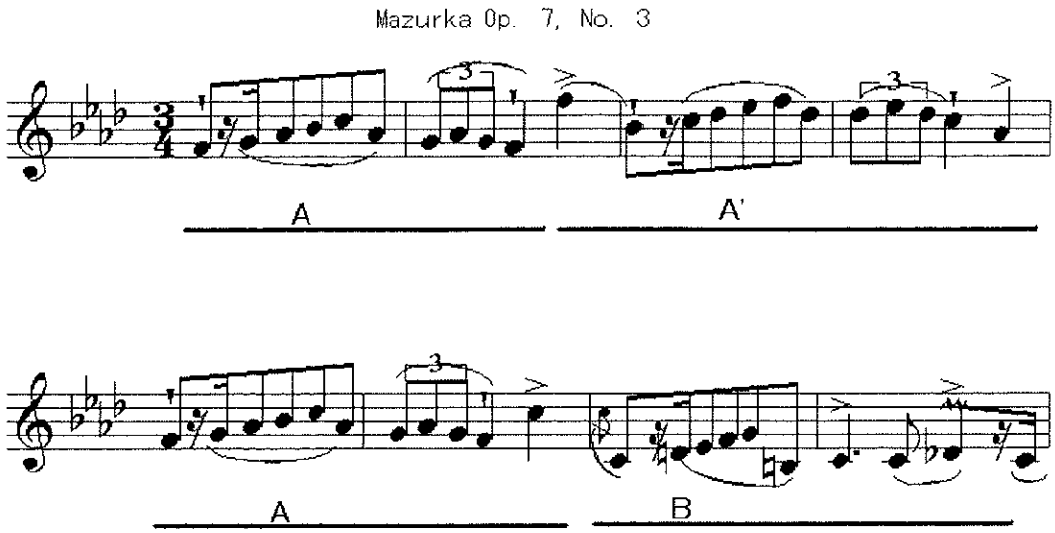


Figure 3.2: Musical structure of Mazurka Op. 7, No. 3, by Chopin used for performance synthesis —ScoreMaker

3.3.2 Accompaniment systems

We had implemented a few basic but interesting systems which may let people who are unfamiliar with music enjoy rendition of themselves. A system gets instruction for tempo and dynamics from two keyboards on which a player manipulates key. Another system gets a portion of performance from keyboards where a player wants to play or can play, then tempo and dynamics are calculated from the input to control the succeeding part of the music.

Introducing musical structure to accompaniment systems have solved elegantly several difficulties to have improved the system output. For the preparation of accompaniment data, the expression refers the way a human pianist plays the melody of the corresponding structure.

- Accompaniment for an acoustic grand piano.

  The primo by a human player is accompanied automatically by the computer controlled second. Before the actual ensemble, performance by human player is rehearsed to find out the basic tempo elasticity and

to generate performance of the second by imitating expression of the rehearsal data.

During ensemble, the system detects the tempo of human performance then anticipates the player's tempo and the beginning of the next structure (measure, motif, phrase, or sentence) [IOD+96][ITMH93][MI95] [ODI96].

As described in Section 3.2, a difficulty of an accompaniment system is caused by the mechanical specification of the grand piano that it must receive MIDI data 500ms prior to the actual key movement.

- An orchestra accompaniment to an acoustic grand piano.

  An orchestra accompaniment system provides the automatic tempo control facility, so that a pianist can practice or enjoy piano concerto by oneself.

- Ensemble with triggers.

  A drum set with MIDI I/O accompanies an acoustic piano played by a human, or, conversely, the acoustic piano automatically follows the percussion played by a human [OI95]. This system is the only exception which didn't use score information provided in advance but only uses triggers from the MIDI I/O device.

  Applying it to "An Der Schönen, Blauen Donau Op. 314" by Johann Strauss II, in which tempo changes in larger scale at some points, has shown the necessity of giving score information, data of musical structure, and rehearsal data to the system for better performance.

3.3.3 Performance visualization

We have developed several types of performance visualization systems in Psyche all based on musical structure [HIM96a][HIM96b][HIM97a][HIM97b] [IHIM96]. All of them have proved that the performance characteristics are recognized when performance is presented visually based on musical structure. They get MIDI data and the data on musical structure, such as the information of motif or sentence, as their input. The detail of an integrated performance visualization system will be described in Chapter 4.

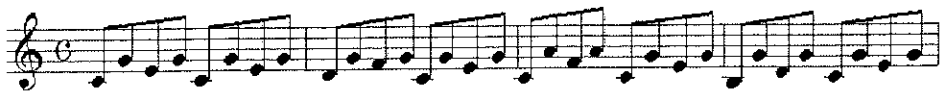


Figure 3.3: Alberti bass from Piano Sonate KV 545, C major, by Mozart —ScoreMaker

Though a system has not been implemented yet, we have analyzed performance data on agogics and dynamics to derive some interesting results [Yag97]. Example observations on several sample performances by professional pianists are as follows.

- The higher notes in Alberti bass[1] (Figure 3.3) played by the left hand are played louder and longer than other notes, though not recognized by listeners, moreover it is contrary to the player's intention.
- A note in the accompaniment part is played softer than other notes when two or more keys are pressed simultaneously.
- Usually a note in the melody part is played first when two or more keys are expected to press simultaneously.

These will be usable in synthesizing and tuning performance.

[1]from [Ken98]: simple (and often commonplace) accompaniment to a melody, consisting of "broken chords", viz., broken triads of which the notes are played in the order: lowest, highest, middle, highest. It takes its name from the Italian composer who favoured it, Domenico Alberti.

Chapter 4

Performance Visualization

Music, whose information is thought to be obtained through listening, can also be understood through viewing. As some researches have made use of the complementary functions of vision and hearing [BWE94], so as information through hearing is not the only object to be manipulated in computer music research. There are reports on the importance of visual feedback for acquiring technical skills of performance notably in shorter time [Slo82]. They are as follows.

- A computer based interactive aid for musicians that allows a passage played on the keyboard of an electronic organ to be displayed visually in a modified notation that reflects the exact duration of the notes played. With the aid of this feedback, a pianist of concert standard improved his performance of a triple trill quite dramatically within minutes.

- With visual electromyographic biofeedback, wind players learned within minutes to suppress or activate specific parts of the buccinator muscle in the cheek. Singers can also be helped in a similar way.

A musical performance visualization system based on musical structure has been designed and implemented in order to analyze and synthesize musically expressive performance data.

Although sequencer is a widely pervasive software for generating musical performance, it is not the most appropriate software for music rendition especially for classical music, because it does not pay much attention to music interpretation and the generating process from a score to its performance, where the concept of musical structure plays an important role.

Our system gives performance analysis uniformity and objectivity in terms of musical structure, in addition to confirmative and quantitative description by performance visualization. The system allows users to edit performance data interactively. Some tools based on musical structure and some constraint functions are provided here in order to prevent the edit of data from destroying musically plausible performance, or even positively give data musically impressive and artistic expression. The edited data are performed with the display and animation of a visualized figure. The system was evaluated on the basis whether it is usable to generate expressive musical performance data. It has received good evaluation for functions particular to the music editor, while some problems on usability have been pointed out.

In Section 4.1, the existing systems in the market and research for performance visualization and editing are described. In Section 4.2, a special point in the design of Psyche's visualization system and its usage are described. The evaluation is shown in Section 4.3.

4.1 Music Editor

There are many music editors both in the market and as research tools [Roa96] which can be categorized to two groups: one is to manipulate mainly score (music printing) and the other performance information (sequencing).

The purpose of music printing is to make a pretty musical score. Score can be obtained by transformation of MIDI performance data, bitmap file of optical recognition, pasting note symbols onto a score, or a score description language. The score in Figure 4.1 (the same as Figure 3.2 in page 29) is drawn by pasting note symbols onto a score with Kawai's ScoreMaker2.0 [Kaw98]. The software has also the function to automatically generate score sheets by optical recognition. The score in Figure 2.1 (page 8) is the same sample score drawn with MusicTEXversion 5.03 [Tau94].

Sequencer is explained by Roads as follows [Roa96].

> A sequencer is a type of recording and playback system with a programmable memory. ... A sequencer can be implemented in any of three forms:
>
> 1. A software package running on a general-purpose computer.

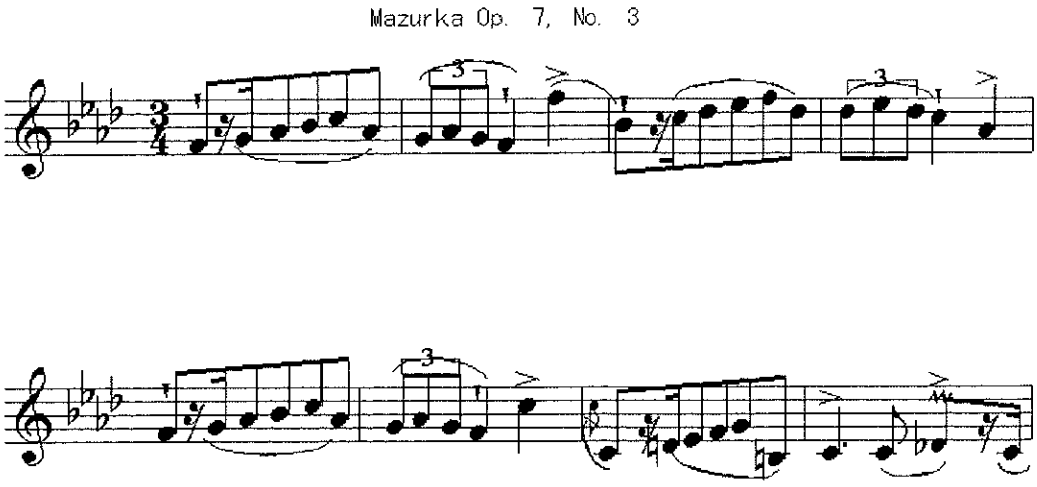


Figure 4.1: Mazurka Op. 7, No. 3, by Chopin (right hand) —ScoreMaker

2. A dedicated box with buttons and knobs for entering, editing, and playing back sequences.
3. A robot, more or less in the image of a human performer.

A software package is flexible and has a graphical display, ...

Psyche's performance visualization system is a sequence software for analysis and synthesis of performance. Though sequence software in the market is the most popular music editor to handle performance data these days, it has not considered musical structure which affects performance. For a music editor to handle performance data of classical music pieces, it is desirable, even necessary, that users can understand the process of obtaining performance from a given score through the editor.

A distinctive point in Psyche's music editor is its visualization of performance data based on musical structure. Performance is visualized for the purpose of analysis and synthesis.

In 4.1.1, sequence software in the market and music editor in research are introduced with their performance visualization figures. In 4.1.2, perfor-

mance analysis and synthesis on Psyche's performance visualization system are described.

4.1.1 Current systems

Visualization by sequence software in the market

So far, sequence software in the market has been used mostly for manipulating popular music, not classical music. We have to be careful about the difference of analysis and synthesis of musical rendition between popular and classical music. It is often the case that agogics add musical expression to classical music while they are not so much in concern in popular music.

Though sequence software in the market provides some tools for specifying performance expression such as expression curves, there is no concept of musical structure. The lack of musical structure leads the undiscernibleness of the expression rationale and the ambiguity of relationships between a part of musical score and the corresponding performance. In order to render music with an existing sequence software, we have to specify musical expression from the very beginning to the very last one by one which is not a simple task and gives difficulty in giving performance a consistent interpretation. The local specification of expression can be harmful to the overall rendition.

Figure 4.2 is a *score window* of Yamaha's XGWorks 2.0 [Yam98]. The score represents performance of the beginning of the score in Figure 4.1 (Mazurka Op. 7, No. 3, by F. Chopin) by Marc Laforet, a professional pianist (both hands). Although the score window just looks like a score of common music notation of western classical music, it does not have the meaning of the original score on which composers put their musical intentions. Also players cannot understand how to play the piece, even melody, rhythm, and harmony. Namely, in spite of the similar outlook to the common score notation, Figure 4.2 includes no higher level information necessary for generating performance. It does not visualize performance, but MIDI data. Figure 4.3 is a *piano roll window* of the same software. Though users can modify the length of each note in both windows, no one would like to change expression in that way.

Figure 4.4 shows the visualization of the same performance as in Figure 4.2 and Figure 4.3 (from the ninth to the twelfth measures by both hands) using another sequence software, Recomposer [Com].

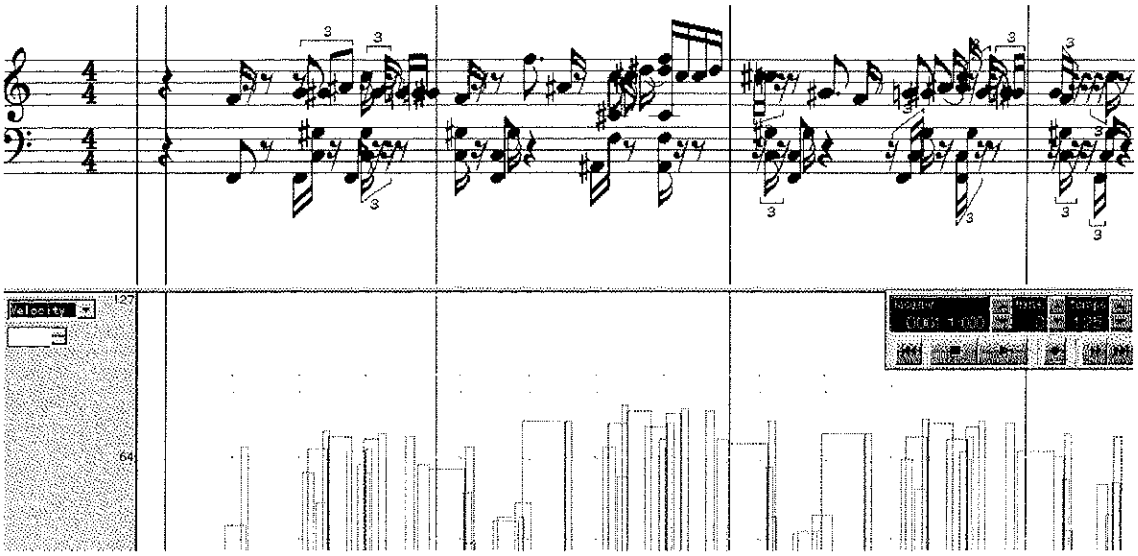


Figure 4.2: Visualization of performance (1): score window of XGWorks 2.0

MIDI events are combined to make note events to be shown on a score. In the bar graphs under the score and piano roll figures, each bar corresponds to a note where a width and a height represent the length and dynamics respectively. Since the concept of musical structure is not involved in the bar graph, we can only see that performance goes ahead along a time line. It is difficult for users to understand the meaning of the performance there.

Music Editor in research

There are some systems built in research for representing performance.

Chafe's system [CO96] represents a score resembling to a common score notation where notes show their performed information: the size of each note head represents dynamics and the distance between the two consecutive notes represents the span. It is easy to compare two musical renditions with this representation. Though it can be used to generate the $n + 1st$ performance from n performances of a piece, it cannot generate the performance of the different piece.

Honing emphasized the relationship between expression and musical structure [Hon92]. He mentions that performance expression is derived not from

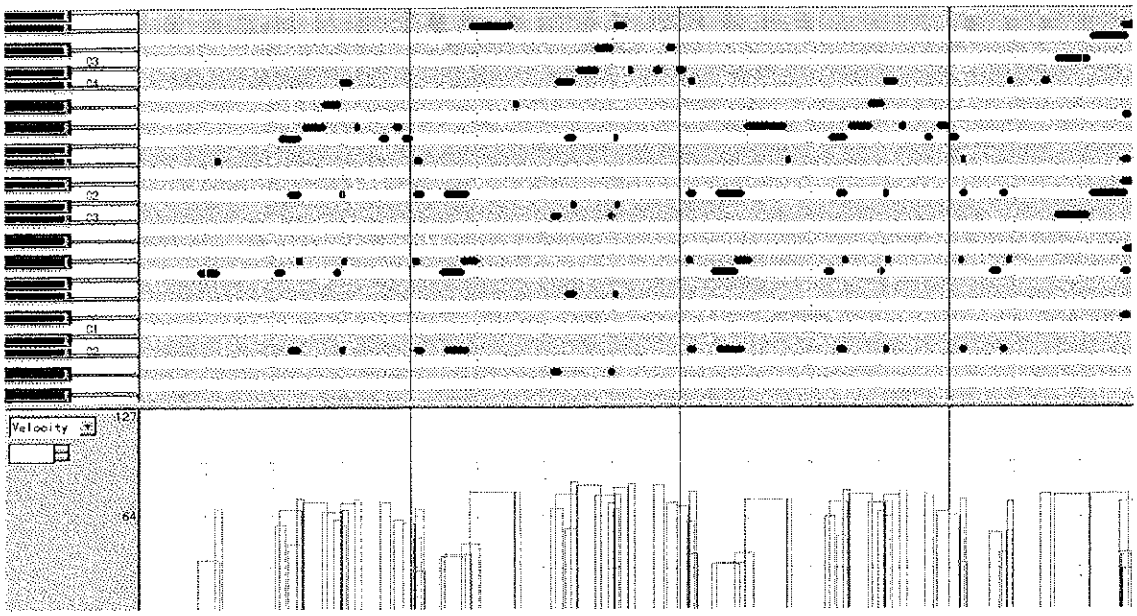


Figure 4.3: Visualization of performance (2): piano roll window of XGWorks 2.0

the standard performance based on an existing score, but from the difference of performance of an occurrence of musical structure. Concerning that expression affects each other, his system, Expresso, introduces a restriction to change in a portion of performance in order not to give damages to expression based on musical structure and its hierarchical structure. He built the system by CLOS and called it the next generation sequence software with GUI.

Oppenheim's DMIX [Opp92][OW96] is a tool to represent performance on a score where a composer can specify his (her) own expression on composition. The system tries to enhance the representation power of the score notation.

RUBATO by Mazzola [MZ94] modeled the performance by a human with the four stages: score (input), musical analysis, music interpretation, and performance (output). He points out three issues in synthesizing performance:

1. make representation both for symbols and semantics on a score.

2. actualize the physical representation of the generated performance.

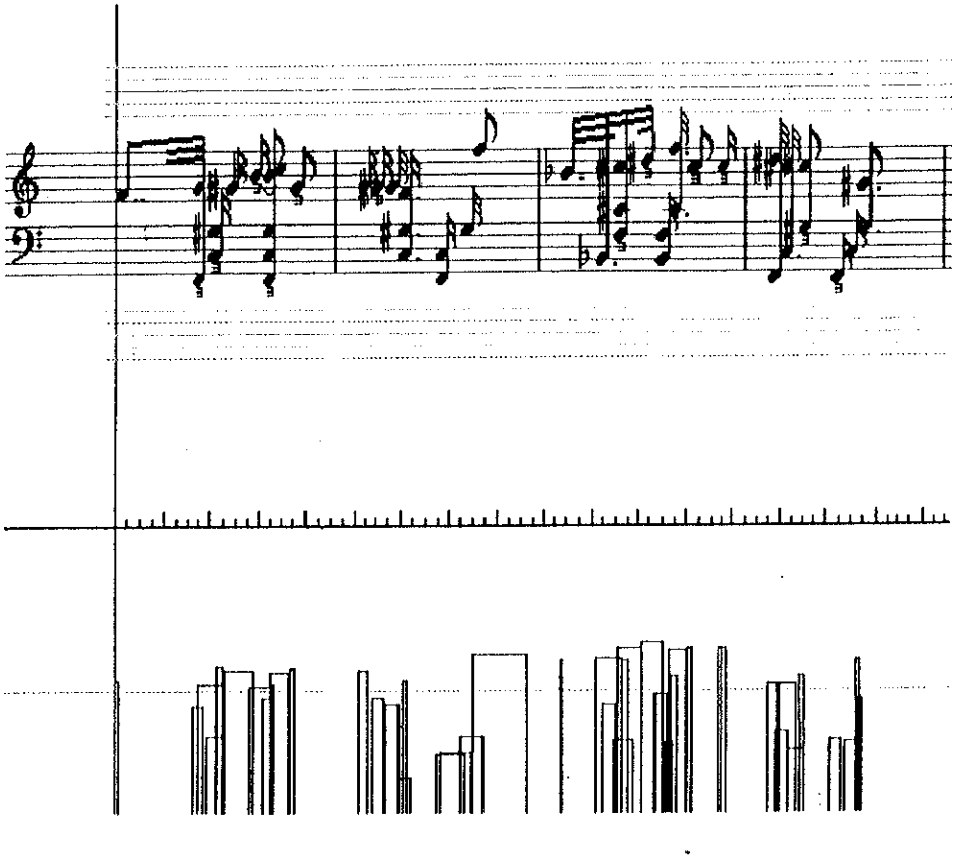


Figure 4.4: Visualization of performance (3): Recomposer

3. build a mapping methodology from a score representation (1.) to physical representation (2.).

A notable point in RUBATO is to introduce *performance score* which exists between the traditional score of common music notation and performance. (Performance score is hard to understand intuitively though.)

Degazio made an editor [Deg96] which is based on the Generative Theory of Tonal Music by Lerdahl and Jackendoff [LJ83] to visualize its metrical structure. The purpose of the editor is not clear whether to be used for performance synthesis or only for analysis assistance.

Currently, research of performance synthesis and the environment (music editor) for it are independent each other. It can be commonly said that good software environment is vital to excellent research of computer music as well as it does to any other research areas. We proposed a music editor for the research of performance analysis and synthesis.

4.1.2 Performance analysis and synthesis on Psyche's music editor

Performers apply performance rules (or they should be called common sense of performance as they usually use it unconsciously) which are independent to the concrete sequence of notes. Agogics and the change in dynamics appear based on musical structure, because the rules to cause expression are context-free based on musical structure and relationships between occurrences of musical structure. Sloboda shows how human performers use context-free rules [Slo82] in their musical rendition. Widmer [Wid93][Wid94] and Balaban [Bal92][Bal96] also point out that those rules are based on musical structure from the view point of researches in music cognition, computer music, and artificial intelligence. It is known that musical rendition constitutes hierarchical structure, hence the higher level structure affects lower level in performance [Bal96][BDCM93][Deu87].

We concerned these musical facts in the design of the music editor of Psyche. The following is the purpose and strategy of the editor for both the analysis and synthesis of performance.

- **performance analysis**
  - Analyze performances by more than one performer of more than one musical piece.
  - Analyze musical structure of the target musical piece on which performance is synthesized.
  - Derive context-free performance rules based on musical structure.
- **performance synthesis**

  Apply performance rules to the musical structure to synthesize performance. Performance should be synthesized considering the following items.
  - Relationships between occurrences of musical structure of the same level make the consistent performance.
  - Performance of the higher level affects the lower to make the deep performance.

4.2 Performance Visualization System

4.2.1 The design principle

Following is the design principle of the music editor where performance data is more easily analyzed and synthesized based on musical structure. The obtained performance should not only be a plausible one but rather artistic and expressive.

- **Performance Visualization**
  - Each performed note is not isolated but has its own role in the performance in the relationships among notes. The visualization can show the move of music along time and the relationships between occurrences of musical structure as well as those between notes.
  - In order to support (complement) the understanding of music through hearing, the visualization can clarify the relationships between the performance expression and its position on a score and musical structure.
- **Performance Analysis**

  In order to analyze relationships between musical structure and performance objectively, qualitatively, and quantitatively, the following items should be concerned in the music editor.
  - Multiple ways of visualization.
  - Comparison of several performances on a figure.
- **Performance Synthesis**

  For synthesizing performance data, the following should be concerned in the editor.
  - Applying performance rules to synthesize the first version of the performance using data of score and its musical structure.

- For refining the synthesized performance, tuning functions are included. The tuning functions should care that the overall performance be musically consistent by avoiding the harmful local tuning.
- The confirmation of the synthesized performance is through hearing and visualization.

4.2.2 Examples

Using the sample score in Figure 4.5, sixteen measures, from the ninth to the twenty-fourth, of Mazurka, Op. 7, No. 3 by F. Chopin, the functions of the music editor are shown as the realization of the design principle described in 4.2.1.

The score consists of two sentences each of which consists of four motifs[1]. These sentences resemble in the rhythm, the melody, and the pitch of notes in each sentence. The second and the third motifs in each sentence are regarded resemblant to the first motif. The pitch of each note in the third motif is the same as that in the first motif, while the second is about four degrees higher in the average than the first. These relationships are shown in Figure 4.6.

Visualization (1)

Figure 4.7 shows visualized performance of the right hand of the sample score in the music editor. We use the performance by Marc Laforet from a floppy disk for YAMAHA piano player (Marc Laforet, Chopin Recital YPA-1069). The graph is shown based on musical structure.

Performance visualization starts from the left-most fan shape and goes clockwise (when a half circle is under the horizontal line, it goes to the opposite direction). A sector of a fan shape and a half circle represents a beat and a motif respectively. The contiguous motifs are shown in the opposite site of the horizontal line for the continuity of music. Also users can compare the performance of occurrences with this conversion of the angle. In Figure 4.7, a radius represents the average value of the span between notes in each beat. There is another mode to show the value of dynamics to a radius.

[1]We mentioned an occurrence of sentence consists of eight measures and that of motif two measures (in Chapter 2, page 10). In 4.2.2 *Examples*, "motifs" and "sentences" are used for occurrences of motif and those of sentence respectively.

A point in this figure is to show the information of each beat, not each note, based on musical structure. It is for the purpose to understand the more global characteristics of performance, such as the relationships among performance of occurrences of musical structure.

From Figure 4.7, we can understand at a glance that two occurrences of sentence resemble in performance where they do in musical structure. Also the first and the third occurrences of motif are played similarly. Performance analysis through visualization tells the existence of a context-free rule that two occurrences in resemblance are played similarly.

Though the first and the second occurrences of motif also resemble in musical structure, the sectors for the first and the last beats are shown differently. It is because the performance of a phrase, which is the higher level musical structure including two motifs, affects the performance of the lower structure. We can explain the phenomenon with the application of a context-free rule on a phrase that the first and the last note of a phrase is played longer than other notes. Then the performance of the phrase is inherited to a motif of the lower level. We can confirm by visualization the musical interpretation of the higher level affects the lower level.

Figure 4.8 shows three performances in two visualization modes. Three performances are, from top, the compiled score data written in Europa, by Marc Laforet, and synthesized one with rules regarding relationships between occurrences of musical structure (R2 in page 28 is an example). The radius of a sector in the left three is for dynamics, the right for span as in Figure 4.7.

Performance data generated from an Europa file are visualized as all radii are the same in the right because there are no indications for agogics in the score (accent symbols make the radii differently in the left). The figure can explain that the automated performance from score information gives listeners mechanical and monotonous impression. By comparing performances by compiled Europa and by Marc Laforet, we can find performance artistry especially in the right figure (representing agogics). Performance based on musical structure which gives listeners impression is shown as the difference in radii, especially in the beginning and at the close of musical structure in some levels.

We can see the resemblance of figures between Marc Laforet's performance and synthesized one. Actually the synthesized performance was highly evaluated by a professional musician which proves the visualization in this shape can tell the musical plausibility.

Visualization (2)

Visualizing performance in different figures provides the easy understanding of performance from several view points. For obtaining the more local information of each note or the relationships among notes are visualized as in Figure 4.9.

Figure 4.9 shows the first four measures, namely a phrase, in the sample score. The performance starts from the top of the circle then goes clockwise. Each colored sector represents a note whose color, radius, and angle show the pitch, dynamics, and length respectively. We can see more clearly the legato and staccato in Figure 4.9 than in Figure 4.7 . On the other hand, it is not possible to see the expression derived from the relationships between occurrences of musical structure.

Visualization of comparison

There are two types of performance comparison.

1. Compare different portion of a musical piece played by a performer. By this comparison, we can see how the repeat or the variation of a theme is played by a human performer.

2. Compare the same portion of a musical piece played by different performers. We will see the performance similarity there to derive context-free performance rules.

Figure 4.10 shows the comparison of two performances by two players based on Figure 4.7. A solid line represents performance by Marc Laforet and a dotted line by another professional pianist.

We can observe performance characteristics on this piece.

- Common ground in their performance.
  - Both of them play longer at the beginning and the end of a sentence.
  - Occurrences in resemblance are played similarly. Both pianists play the first and the third occurrences of motif similarly.
- Differences in their performance.

- The third beat in many of the motifs is played differently. A player plays it longer than adjoining beats, while the other plays it shorter.

Performance editing

There are two ways for editing visualized performance. One is to click and move a mouse on a sector in Figure 4.7 in What You See Is What You Get (WYSIWYG) way. The other is to use some of the functions which are for performance generation, giving performance consistency, and avoiding damages by editing. They are as follows.

- Generating performance data by applying performance rules to data for an occurrence of musical structure.
- Applying expression curves [ITC+95] regarding agogics to occurrences of musical structure.
- A function to constrain change of performance to keep the musical meaning. An example is a constraint in the value change of a performance parameter. An extreme change over a criterion is not accepted.
- A function to automatically pervade the change to an occurrence to other occurrences in relation to keep the overall musical meaning.

The modified performance can be confirmed by hearing synchronized by the animated figure.

The music editor was designed with object-oriented framework and implemented on Free BSD using gcc and OSF/Motif [HIM96b][HIM97a][HIM97b] [IHIM96].

4.3 Evaluation

We evaluated the music editor whether it is useful for generating expressive performance. Four subjects are asked to give answers to a prepared questionaire and comments. Subjects are student members of the project Psyche. They represent two factors of

1. having good experiences especially in classical music, and

2. having used a sequence software.

Two of them engaged in generating performance data.

They are shown visualized performance in the style of Figure 4.7. The performance is generated by applying performance rules to the data of the sample score in Figure 4.5 and its musical structure. Because the sample score is often used in the project Psyche, all four subjects knew the music well. Then they are explained how to use the music editor and asked to generate performance on it. There is no restriction in time for using the editor.

4.3.1 Results

The subjects answered the prepared questionaire in five points where 1 is the strongest NO and 5 is the strongest YES. Table 4.1 shows the questions and answers with average points. Points in the parentheses are calculated using two subjects who work on performance synthesis. Those who engaged in the research of performance synthesis gave the better evaluation. An interesting point is that two groups (one is engaged in the research of performance synthesis, the other is not) give the opposite evaluation in constraint functions for performance editing (the question in the table is: Importance in constraining on performance data editing). Those who worked on performance synthesis evaluated it useful while others didn't. Functions for performance editing are more reflected by musical knowledge than superficial value editing.

Two students who have used a sequence software were asked to give comments on comparing the music editor and the sequence software. Showing information of both the length and the dynamics of performed notes was a point of discussion. Though there is no conclusion on this, an opinion is the multiple information is useful, while the other is one information is good enough if performance is visualized based on musical structure. Colors can be used to represent performance information though they are not used in Figure 4.7 currently.

Required functions are pointed out by subjects as follows.

- **Usability**

Questions	Results
Visualization of Figure 4.7	
Ease of understanding	3.5(4.0)
Inconvenience in showing either length or dynamics	2.5(2.0)
Tools for Performance Analysis	
Value in Figure 4.9	3.3(3.5)
Understandability of performance comparison by layering (Fig. 4.10)	3.0(4.0)
How to synthesize performance data	
Usefulness of expression curves	3.3(3.5)
Usefulness of performance synchronized by the animated data	4.8(4.5)
Usefulness of handling left button of a mouse	4.3(4.5)
Importance in constraining on performance data editing	3.3(4.5)

5 4 3 2 1
Strong YES ← — → Strong NO

Table 4.1: Evaluation of the music editor (questions and answers)

- Range of selection. Currently only a sector of a fun shape can be selected at one time. It is desirable to include the selection of other units such as motif, phrase, and sentence.
- The pervasion of performance change. Currently performance change in seed occurrences is transferred to the related occurrences. It is desirable to include the cut-and-paste function which enables the pervasion of change to other portions of a score that are not necessarily related in the sense of musical structure.
- Change by value. Currently, the mouse handling is the only way to edit performance. It is desirable to change performance by specifying the absolute or relative value of MIDI data. The numerical value is also useful to see the result of the change.
- Show scale on a figure.

- **Functions for performance editing**

- Specify the change of smoothness of the performance.

  It is desirable to specify *ritardando*[2], for example, with the range and the difference (or curve).

- Change inside a unit.

  We have to be careful in changing performance value related to tempo especially when the modified range includes symbols for agogics and length. For example, if a range, where notes with *staccato*[3] are, is modified to play slower in the editor, then how to play staccato should be specified by the modification of each note inside a unit. The uniform modification is not acceptable.

4.4 Summary

Visualized performance in the style of Figure 4.7 is evaluated to be easy for understanding performance rationale because it shows the correspondence between the musical structure and performance. Visualizing the comparison of performances is useful in the analysis and synthesis of performance.

The music editor based on musical structure described in this chapter was evaluated valuable in the performance analysis and synthesis. On the other hand, the usability has rooms to be improved not only concerning the comments by subjects but also consulting functions required in music editors described in Roads [Roa96].

[2]from [Ken98]: hold back gradually.

[3]from [Ken98]: detached. Method of playing a note (shown by a dot over the note) so that it is shortened—and thus "detached" from its successor—by being held for less than its full value.

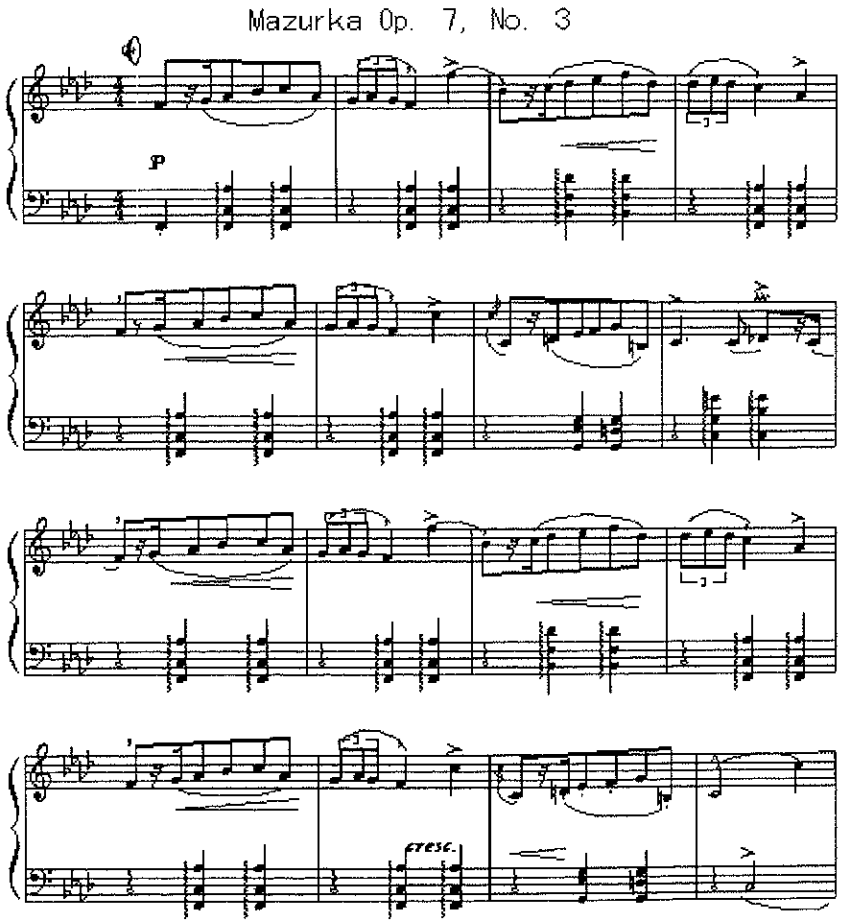


Figure 4.5: Mazurka Op. 7, No. 3, by Chopin (both hands) —ScoreMaker

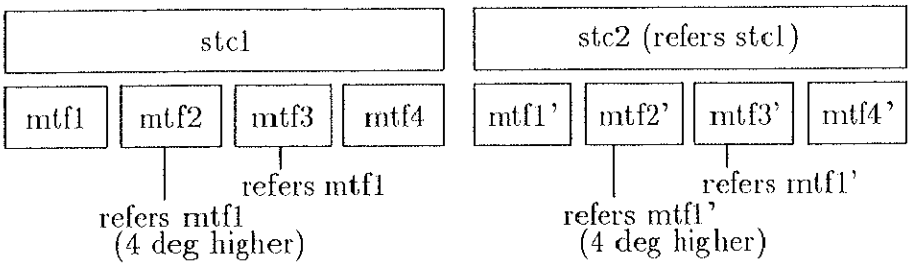


Figure 4.6: Musical structure (hierarchical structure and relationships between occurrences)

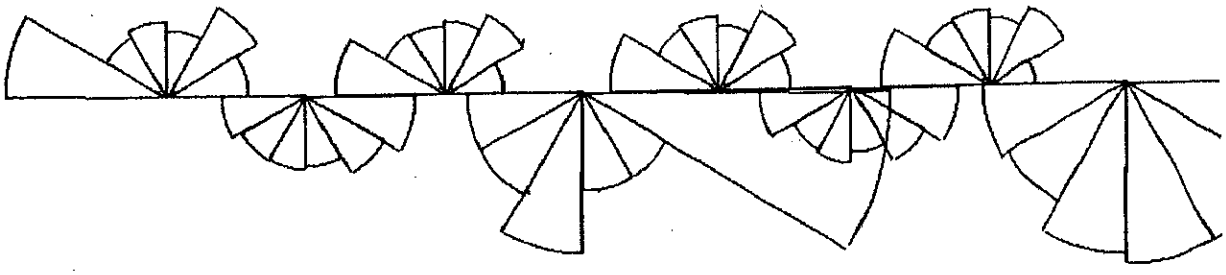


Figure 4.7: Visualization of performance (4): Psyche 1-1, by Marc Laforet (time span for a radius)

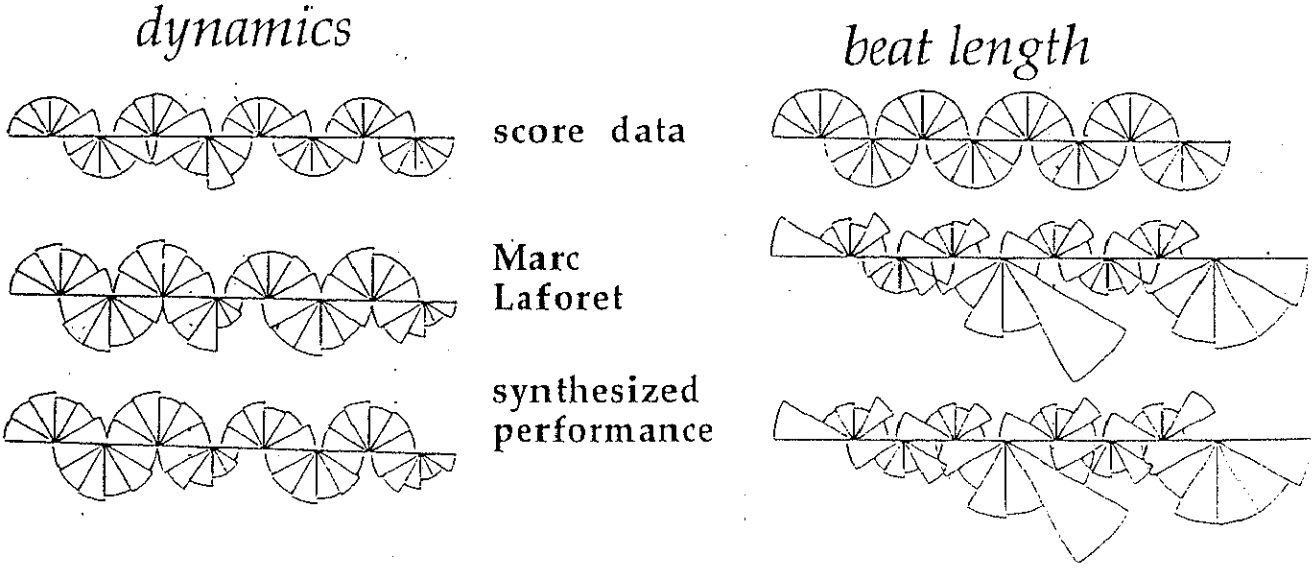


Figure 4.8: Visualization of performance (5): Psyche 1-2, by three players

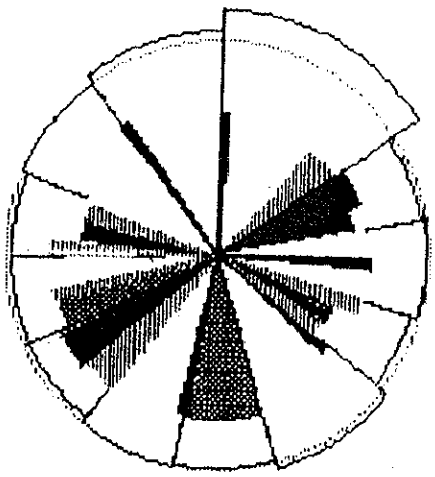


Figure 4.9: Visualization of performance (6): Psyche 2, four measures

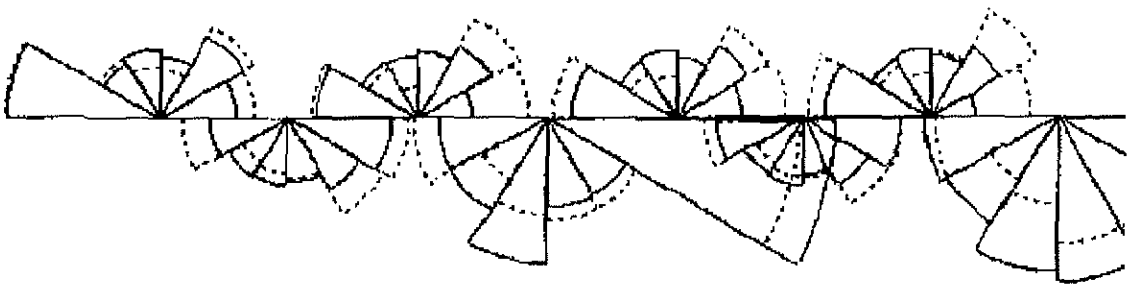


Figure 4.10: Visualization of performance (7): Psyche 3, comparison of performance by two pianists

Chapter 5

Musical Analysis System Daphne

A computer-assisted musical analysis system Daphne—Declarative Analysis of PHrasing aNd Expression— has been designed and implemented in order to analyze musical pieces for performance synthesis [HIL97][HLI99]. Musical structure is the primary item in the analysis.

The output of musical analysis, its name and format, in musicology is far from being standardized from computer science point of view. For example, there are ambiguities in analysis terms or sometimes there is no term for a musical phenomenon which people can understand and discuss though. To some extent, music can be represented mathematically. On the other hand, we are confronted in actual musical activities by the unavoidable complication and ambiguity which should be solved in building musical analysis systems. This leads to a fact of computer music systems for musical analysis: there is no one in the market and only a few in research.

Daphne enables users to obtain and share analysis information which implicitly but innately exists both globally and locally in a score. It is musical structure and its attributive information. Analyzed information specifies a portion of a score on which performance rules of Psyche are applied.

It is more important to obtain analysis information of the most suitable for performance synthesis than the complete automation of analysis. Daphne is designed as a computer-assisted performance analysis system which users can autonomously use or ask to automatically provide analysis candidates.

In Section 5.1, the existing musical analysis systems are described. Then

in Section 5.2, Daphne is described in its design along implementation and a sample usage. Two issues in building musical information systems, how to evaluate them and the possibility of using existing software methodologies for them are discussed in Section 5.3.

5.1 Musical Analysis System

5.1.1 Current systems

Though there are only a few musical analysis systems, we can classify them from the purpose and the basis of the systems (a theory on which the system is built). The purposes are as follows.

- Computer-assisted musicology. Computing power enables musicologists to analyze the huge amount of data automatically in the shorter time.
- For performance synthesis. Using information of musical analysis for performance synthesis.
- For proving a theory. A musical analysis system is used to attest a musical analysis theory.

The theories on which systems are built are as follows.

- Musicology in general. Chord progression, structural analysis, and other general analysis items are introduced from musicology.
- *Leitmotiv* (leading motive or representative theme)[1], for a special purpose analysis.
- The more recent computer and cognition related theory such as GTTM [LJ83] described in Chapter 2.

Table 5.1 shows the purpose, theory, and analyzed items of four systems in research.

[1]from [Ken98]: to describe a short constantly recurring musical phrase or theme denoting a person, thing, or abstract idea. Composers throughout history have used the device in one form or another, but it was raised to its highest and most complex form by Wagner, especially in *Der Ring des Nibelungen*, where the subtle combinations of Leitmotiv create symphonic textures.

System	Purpose	Theory	Analyzed Items
(*Humdrum based*)	Leitmotiv analysis		Leitmotiv
	evaluate IR model	IR model	implicative intervals, notes to follow the interval
Apollo	unknown	unknown	melody, rhythm, harmony, statistics on diversity of pitch, etc.
Rubette	performance synthesis	GTTM, musicology	metrical structure of GTTM, motif, melody, harmony
Uwabu's tool	resolve the structural ambiguity of GTTM	GTTM	prolongational tree

Table 5.1: Systems of performance analysis

Humdrum created by Huron [Cen] is a toolkit consisting of a set of musical analysis of Unix-like commands. In spite of the line commands, the expected users are music oriented people, especially music theorists or music analysts, rather than those involved in creative musical activities. It provides a wide range of analysis. Examples are as follows (excerpts from [Hur]).

- In Urdu folk songs, how common is the so-called "melodic arch"–where phrases tend to ascend and then descend in pitch?
- Which of the Brandenburg Concertos contains the B-A-C-H motif?
- Which of two English translations of Schubert lyrics best preserves the vowel coloration of the original German?
- After the V-I progression, which harmonic progression is most apt to employ a suspension?

A research [Kor95] used Humdrum to analyze Leitmotiv of the whole piece of *Die Walküre*. So far it was not easy to analyze completely the piece by human power because of the length. Humdrum returns Leitmotiv of the whole piece from score data and entries of Leitmotiv as input. By providing the GUI of the original score, the system tried to be acceptable by professionals of music.

Another research [TS95] used it to evaluate Implication-Realization model (IR model) [Nar92]. IR model tries to explain the melodic structure by the implication of innate principles and the manner in which they are fulfilled or denied. Humdrum was used to identify implicative intervals in chorales by Bach and melodies by Schubert and has shown that implicative intervals following the IR model could introduce the next notes.

Though Apollo project [Poo95] is described by a professional pianist and musicologist, the purpose of the system was not clear from the description.

Rubette [MZN95] is a musical analysis system consisting of a set of analysis modules in a musical-performance software platform, Rubato[MZ94]. Thus the output of Rubette is to be used for performance synthesis. Currently metrical and rhythmical analysis based on GTTM, and motif, melodic, and harmonic analysis in the sense of general musicology can be analyzed in Rubette. It also includes GUI for better and easy understanding for the wider range of users.

Uwabu's tool [UKI97] is a part to construct a musical interpretation model as a computational model in the project at Inokuchi lab., Osaka University. The model is for the synthesis of musical performance with expression rules that are automatically extracted from information of musical structure. Rules are applied to another piece by searching the relationships of occurrences of musical structure. Since his tool is based on GTTM, it generates relationships of occurrences in prolongational tree with note and chord.

5.1.2 Musical analysis by Daphne

The purpose of musical analysis through Daphne is to provide information which is useful for performance synthesis. Daphne should avoid ambiguities in representing musical activities by human as well as to keep flexibility in interpretation of analysis. Daphne doesn't follow any of a recent analysis theory including GTTM, but is based on the general analysis in musicology. Thus analyzed items in Daphne are commonly used by professional musicians.

In order to keep the quality of analyzed items, Daphne should not be used only by computer science experts but also by professionals of music. Therefore the user interface plays an important role for Daphne to be pervasive among the wide range of users. In order to introduce the reality of musical analysis into the system, the score image and the actual performance sound are involved. Command to Daphne system is entered through GUI, where

users can analyze music as if they did it with a pen, score sheets, and a piano or other instruments in the real world.

Daphne is designed as a *computer-assisted musical analysis system* by making good use of computation in order to add values of analysis automation to realistic analysis.

5.2 Daphne

5.2.1 The design principle

One of the points in designing a musical analysis system is the choice of analysis items for performance synthesis, since those items should be objects to which performance rules are applied.

The way to choose them are either to refer performance rules or to select objects which have musical meanings themselves. We choose items in the latter way because there are not many performance rules currently, and however many rules there were, it can never be said that the rule set is complete (the indefiniteness of musical knowledge described in page 3).

Then we should decide the research basis to choose items. Some computer music systems refer famous theories such as GTTM [LJ83] or the rhythmic structure [CM60]. On the other hand, some researchers point out the pros and cons of these theories to computer music systems [KT94] and even an affirmation exists that no systems can be built leaning on a theory [Bal96]. Therefore we refer books written by musicologists [Ber87][Kum95][LO94] [Wal87], then choose items as follows.

- **musical structure**

  Though not explicitly denoted on the score, musical structure is innate to music which affects greatly on performance [Hon92]. The effect of musical structure on performance is widely accepted by many researchers as described in Section 2.2 (pages 13 through 14).

  Musical structure constructs a hierarchy, in which the name and the size of each layer is not uniformly defined in the music area. Daphne adopted Riemann's musical structure (Section 2.1) where motif, phrase, sentence of two, four, eight measures are basic units to constitute hi-

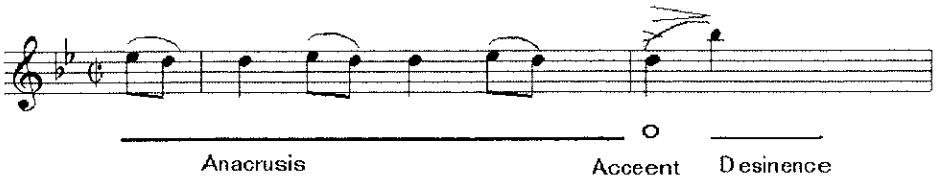


Figure 5.1: Metrik example from Symphonie KV 550, G minor, by Mozart —ScoreMaker

erarchy. Analysis information below inherits the hierarchy of musical structure.

- **Metrik (metre[2])**

  With musical structure, we can understand globally the construction of a piece. It is also necessary the local understanding of a piece, namely the role of each note in an occurrence of structure for performance. Anacrusis[3], accent, desinence are specified as items of Metrik. Figure 5.1 shows an example.

  Metrik plays an important role not only for performance synthesis but also for other area of computer music such as music cognition and beat tracking.

- **relationships between occurrences of musical structure**

  It is useful to clarify the relationships between occurrences of musical structure for synthesizing performance expression [Hon92][H95].

  There is a notation in musicology to represent music style by the similarity among components such as "*ABA*" where the third resembles

[2]Sometimes Metrik is used as described in [Ken98]: "term regular succession of rhythmical impulses, or beats, in poetry and music. Rhythm is no longer accepted as a sufficiently precise definition, metre being considered as the basic pulse and rhythm as the actual time-patterns of the notes within a measure." GTTM's metrical structure represents the metre of the above meaning. While Metrik in Daphne is rather "metrical paraphrasing (psalm)" by understanding the musical contents of a structure as if to versify it. Kumata [Kum95] uses Metrik in the latter sense.

[3]from [Ken98]: unstressed syllable at the beginning of a line of poetry or an unstressed note or group of notes at the beginning of a musical phrase.

to the first. In Daphne, the similarity of melody is specified to the finer structure, such as motif of two measures. The similarity is calculated using the position and pitch of corresponding notes if any, in two occurrences.

- **tonality**[4]

  The tonality is used to confirm the form and correspondences of occurrences of musical structure.

- **harmony and chord progression**

  When a phrase ends with a perfect cadence or imperfect cadence[5], these cadences regulate the time span to the consecutive phrase as well as the close of the phrase.

All of these analysis items are not necessarily specified for the overall music piece. The excess of the specification may be harmful to generate expression. The first five of the analysis items correspond to musical knowledge in Figure 2.3 (page 17) which is implemented into classes of object-oriented method.

Other characteristics of Daphne as a musical analysis system are as follows.

- **computer-assistance**

  Assistant functions are: complementary command input, automatic analysis, a function to propose the next item to be analyzed. Users can either specify or inquire analysis interactively to Daphne.

  The third function gives Daphne the autonomy in analysis, namely the communication is not only from users' command input to Daphne but also the both way including Daphne's suggestions to users.

[4]from [Ken98]: key, meaning particular observance of a single tonic key as basis of composition.

[5]from [Ken98]: any melodic or harmonic progression which has come to possess a conventional association with the ending of a composition, a section, or a phrase. Perfect cadence: chord of the dominant followed by that of tonic. Imperfect cadence: chord of the tonic or some other chord followed by that of dominant.

- **visualization of performance and musical analysis**

  As described in Chapter 4, performance visualization based on musical structure enables users the easy understanding of musical meanings of performance.

  Showing the result of analysis is helpful in confirming and sharing the analysis information among people.

- **special specification for systems of Psyche**

  Psyche's accompaniment systems can generate more plausible output when information reflecting musical structure is given. Examples are the point to accompany and the leading part for occurrences of musical structure. Not categorized in general musical analysis of musicology, necessary information for improving Psyche's systems is also specified through Daphne.

Implementation by object-oriented approach

Daphne has been developed on Windows 95 using Visual Basic for its GUI and Visual C++ for the internal process. Figure 5.2 shows the relationships between classes of Daphne.

Score data written in Europa language is the input to Daphne. A user has to specify a file for a piece written in Europa (Europa file) prior to the analysis. *DEuropa* is a class to handle an Europa file. It loads an Europa file when a Daphne session starts, then generates objects of the class *DMeasure* and *DNote*. When a voice part is specified, a note on a score is uniquely specified with a tuple of the number of the measure it resides and the offset from the start of the measure. An object of DNote represents a note and has attributes of the offset position, pitch, and note value for each note. The values of these attributes are set at loading an Europa file. An object of the class DMeasure corresponds to a measure in a score which has a list of objects of the class DNote in the measure as an attribute.

Daphne's output is analysis information which is handled by a class named *DDaphne*. Analysis information is stored into three types of repository.

- History file of analysis. Users can look back their analysis process.

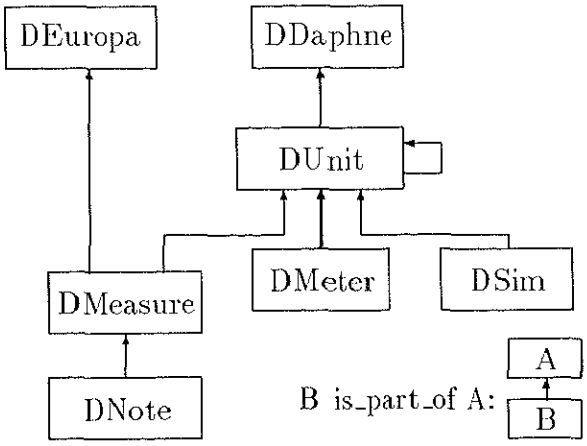


Figure 5.2: Classes of Daphne and their relationships

- Serialized objects which have values specified by analysis. If an analysis session is to continue a previous one, then DDaphne loads serialized objects to restore the end of the prior session.
- UNI formatted file (described in Section 3.1, page 25) for representing note events with musical attributes. UNI format involves information of musical analysis such as whether a note is in an occurrence of motif or it has a metrical accent.

Objects of the class *DUnit* represent occurrences of musical structure. It has attributes for the start and end positions of an occurrence in a score presented by measures and the offset in the measure, the level of the hierarchy (whether the occurrence is a motif, phrase, sentence, or some other), and attributive analysis information involved in the occurrence. Notes in an occurrence of musical structure, namely objects of DNote are obtained through objects of DMeasure. Thus DMeasure bridges the explicit information on a score (measure and note) and implicit information (musical structure). The hierarchical relationship is represented by an attribute of a list of DUnit objects for the upper and the lower level. Also DUnit has an attribute of the previous and the following objects of DUnit of the same level.

The classes *DMeter* and *DSim* are for Metrik and the similarity relationship as analysis items respectively. Each class has different set of attributes for analysis. DMeter has the label which shows the type of Metrik (accent, anacrusis, or desinence) and the position where the Metrik information exists. DSim has a set of a tuple of a referring occurrence and the degree of the similarity. When items of analysis are increasing, corresponding classes are involved in the system.

We will discuss in Section 5.3.2 the possibility and the limitation of using object-oriented method in representing musical knowledge.

5.2.2 Analysis example

Using a sample score of sixteen measures, from ninth to twenty-fourth measure, in Figure 5.3, we will show how to use Daphne for musical analysis. We are implementing user interface for direct specification on a displayed score. Performance for the piece is also available in the interface. Prior to an analysis session, users (or a Daphne administrator, if there were) have to prepare an Europa file, a set of bitmap files of score, and sound files for a piece to be analyzed. Bitmap files displayed in the GUI are made as in Figure 5.4. (All figures in this thesis, indicated by ScoreMaker in the caption, are ps files converted from the screen copied bitmap files.) Each bitmap file corresponds to a page of an original score. Sound files can be UNI format, SMF (standard MIDI file), or WAV (wave file).

Figure 5.5 shows the main window of Daphne. The upper part of the main window is for the history of analysis where command input by a user and the response from Daphne are shown. Responses are prefixed by '%'. The lower part is the command area. A command is generated on the command area according to analysis on a score and recorded in a history file.

Specify a piece to analyze

At the beginning of an analysis session, specify a musical piece to be analyzed by its Europa file name (a command at the first line of the history area).

new maz

Actually the "New" command is one of the "File" menu on the menu bar of the main window. A file is specified and opened through a standard file

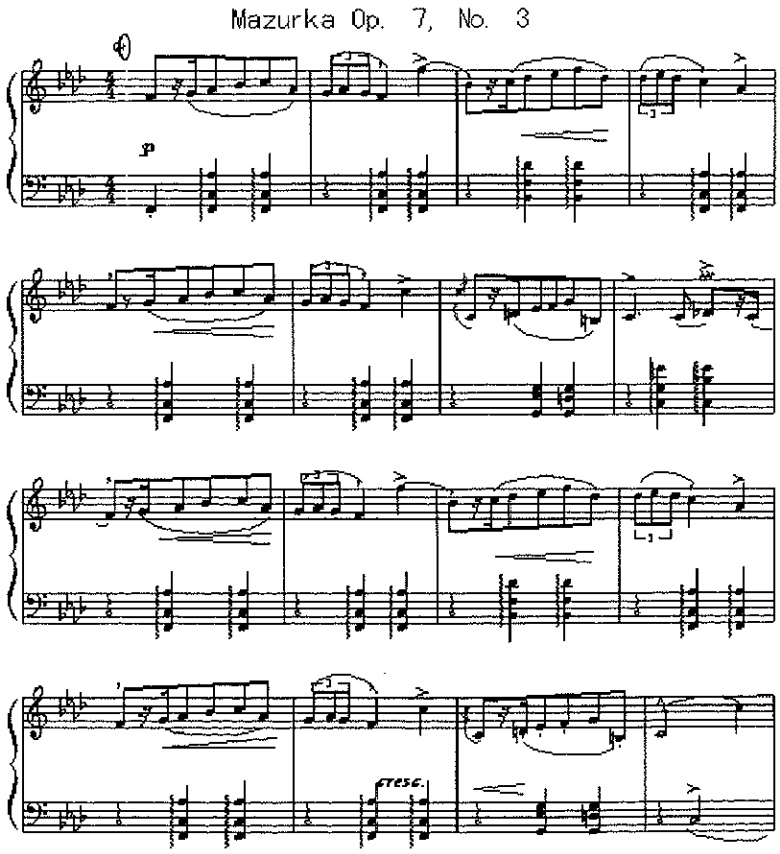
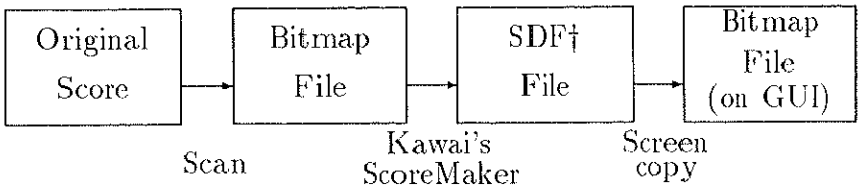


Figure 5.3: Mazurka Op. 7, No. 3, by Chopin (both hands) —ScoreMaker



†: SDF is a file format by Kawai to hold score information.

Figure 5.4: Process to make score images made from original score

list window of Windows95.

Objects of the class DNote and DMeasure are made according to the Europa file.

Inquire musical structure

Opening a score for the piece to be analyzed. Score is displayed as in Figure 5.6.

A user asks Daphne about occurrences of motif in the first half (from measure nine to sixteen) of the sample score. Prefixed by "*show*", the second command in the history area

show motif from 9 to 16

asks Daphne to analyze occurrences of motif. In this WYSIWYG user interface, a user first specifies the range to analyze by the mouse click on the beginning and the end positions, then to specify structure analysis from the popup menu appearing on the score in Figure 5.6. A popup menu includes the same set of commands as "Analysis" menu on the menu bar.

The response to a command is displayed as lines on the score. It visualizes the system reply below which is stored in the history file.

M9(9-10) M11(10-12) M13(13-14) M15(15-16)

It means that Daphne divides the eight measures with the knowledge of "a motif usually consists of two measures" and calls the first motif M9 ("9" for the measure number) as a default name and so on.

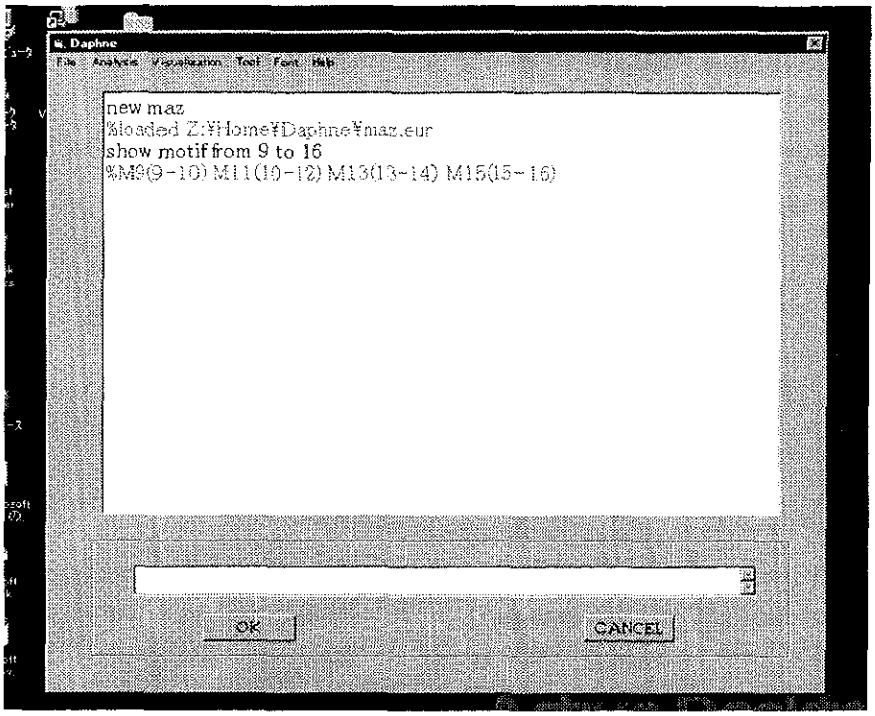


Figure 5.5: GUI of Daphne (1): The main window

Daphne also uses the information of "slur" in the Europa file for more precise and music-oriented analysis. There is a slur from the third beat in the tenth measure (f^2) to the first note in the eleventh measure. It implies that f^2 belongs to the following structure as *Auftakt* (upbeat[6]). The end position of M9 and the start position of M11 are automatically set to the second and the third beat of the tenth measure respectively[7].

Internally four objects of the class DUnit are generated. They have attribute values of each position and the level of the hierarchy (motif).

Sometimes Daphne cannot answer the inquiry of the structure if there are

[6]from [Ken98]: upward movement of conductor's baton or hand, especially to indicate beat preceding barline.

[7]A range is specified by two positions as "M through N", where M and N has different meanings with the same representation of the measure and the offset. M means the beginning the position, while N means the end of the position which is not exactly the start of N, but after the note value length for a note specified at N.

many ties lying across measures. Users, of course, can respecify the musical structure by themselves if they are not content with the result of the inquiry.

Inquire the relationships between occurrences of musical structure

Then a user tries to find the relationships between occurrences of motif just acquired. A command below is generated.

show similarity level from 9 to 16

Since occurrences of motif were analyzed so far, Daphne replies as follows for the relationships between occurrences of motif.

M9: M9(0), M11: M9(3), M13: M9(1), M15: M15(0)

This means that in the range of the inquiry, from the ninth to the sixteenth measure, the first motif M9 is regarded as the standard, then the melody of M11 resembles to that of M9 in a sense, that of M13 is almost the same as that of M9, and M15 can be seen as independent from M9. This is what Daphne interprets the melody in the given range. Number indicates the similarity where the smaller the more similar. Values '1' and '3' as the similarity level in the parentheses don't represent the absolute value of resemblance. In the above case, representing the resemblance qualitatively as $aa''a/b$ has the similar meaning.

Objects of the class DSim are made and attribute value for the relationships of objects of the class DUnit is set by this command.

Specify Metrik

The user may think that there is an anacrusis from the first note of the ninth measure to the second beat in the tenth measure. As for meter, there is no function of inquiry currently, thus a user have to specify the Metrik information. The specification can be done on Figure 5.6 which generates the following command.

anacrusis from 9_1 to 10_2

Internally an object of the class DMeter is made which is involved as an attribute value of the object of the occurrence of motif 9 (M9) of the class DUnit.

So far only objects exist which are specified or inquired by a user. Not all attribute values have been acquired for them but those which were specified or inquired have been set. Therefore, although the note f^2 in the tenth measure is a metrical accent of a phrase starting from the ninth measure (P9) and even a user knows it, no object exists for P9 of the class DUnit nor for the accent of the class DMeter.

By connecting displays on Windows98, we can open more score sheets at a time (Figure 5.7).

5.3 Discussion

5.3.1 Evaluation of musical processing systems

Difficulties in evaluating musical processing systems

It is difficult to evaluate not only musical analysis system but also many other musical information processing systems because of the divergence of realization.

Though some suggestions are proposed for evaluating representation of music [Dan93][WMSH93], they are far from generalized and standard criteria which must be met in order for a representation to be accepted. The objective of the music representation varies: for music synthesis, analysis, composition, printing, and programming. The level of representation also varies: from the very concrete, such as audio signal, to abstract and highly symbolic one which has structural generality with emotional content. Thus these papers could only offer issues to be considered in working on musical representation. (Besides, issues mentioned there are not exhaustive because of the indefiniteness of musical knowledge.) They say themselves that it is impossible to use a standard or criteria for evaluation, though the papers agreed in that structure is a point in designing representation in any objectives and levels.

Needless to say, no clear descriptions on evaluation were made for other research area in computer music. In other words, and if positively referred,

each subject in computer music research is fertile where no standardization could be found nor applied yet.

Evaluation of computer music research and systems have often come up for discussion [Hir97b][Hir97c]. Hirata mentions a reason for few big contribution has made to the computer science research by computer music area as the *infertility* of evaluation system (which is the opposite way of mentioning the fertility of computer music research) [Hir97b].

Before the workshop "Issues in AI and Music — Evaluation and Assessment" attached to IJCAI '97, two panel discussions on evaluation were held. Although the third time, the workshop covered too variant subjects on evaluation to reach a conclusion; from philosophical discussion, methodological, to experimental. Even in the most concrete descriptions on systems in the experimental track, three systems introduced there were performance synthesis [Hir97a], style analysis, and beat tracking. There are no way to compare evaluation method between these systems of different types. We were convinced at the workshop that many researchers on computer music have noticed the importance of evaluation method, still we will have a long way to get evaluation methods for computer music research.

Following is the two main reasons for the difficulty in establishing evaluation methods.

- **divergence**

  There are several research subjects in computer music, besides as described above in representation of music, there are several issues in a subject and many ways to overcome the issue to make the research realize. Thus systems in a category may not be able to be evaluated by a single method.

- **subjectiveness**

  Some computer music research refers *Kansei* [I+94] which was beyond computer system's understanding. When Kansei is a key to the system, subjectiveness is inevitable. For example, the output of performance synthesis systems must appeal to Kansei, like performance by human gives impression to listeners. This requires the new type of evaluation method that should be different from conventional evaluation of computer systems.

The lack in evaluation method may give the following deficiencies to computer music research.

- Difficulty in finding the next step.

  Although some systems apparently have points to be improved, because of the lack in musical knowledge, research survey, or some other reasons, it is not easy to point out which should be refined.

- Make people approve the superiority of a system or even computer music research.

  Think of today's paper refereeing system, the evaluation of a system is required for a paper to be accepted. If the lack of the evaluation method for computer music systems avoids papers from acceptance, then it may affect badly the evolution of computer music research as well as the acceptance by people in other fields.

Evaluation methods of musical analysis systems

As described in Section 5.1 and shown in Table 5.1, there are a few musical analysis systems which were designed and built for their own research requirements and interests. Besides, the differences in their purposes, relying theories, and analyzed items cause the difficulties of comparison and evaluation of systems. The following shows two possible methods for evaluating computer analysis systems.

1. Select subjects both from the system's users and non-users, then evaluate it with the common way of evaluation of other computer systems such as with questionaire.

2. Evaluate each component technique by comparison.

   Though the collection of analyzed items are different among systems, some items are commonly used by more than one system. The analysis method can be evaluated by comparison of those items. For example, the preciseness or the appropriateness of returned value by systems for the relationship analysis can be compared among systems.

Evaluation with subjects The value of Daphne as a computer-assisted musical analysis system for musicology is proved by musicians' acceptance. Professional musicians can advise us the appropriateness or inappropriateness of analyzed values once Daphne is used by them without obstacles. Thus we have been implementing the score-oriented WYSIWYG user interface as in Figure 5.6 for musicians to make use of it. Daphne can be evaluated in its analysis functions and usability in the first method as was Psyche's music editor in Section 4.3, when Daphne's implementation with the score-oriented user interface is completed.

In evaluating systems this way, we have to be careful about how to handle *subjectiveness*. In computer human interaction area, evaluating user interface is assessed by many researchers to establish a research area, though user interface, from manuals to graphical user interface, may include subjectiveness.

The selection of subjects and preparation of questionaire should be carefully done in any areas. We find special characteristics of this subjective evaluation for computer music systems are as follows.

- The variety of users.

  It is necessary to include users of non-computer professionals, especially professionals in music. It can be the case that musicians have the narrower acceptance because of their richer knowledge on music and less on computer than computer professionals.

  When including subjects who are not well-trained in music, then the result of the evaluation is doubtful whether it clarifies the purpose of the system, musical analysis[8].

- The effect to artistry.

  The difference between evaluation of user interface and music systems is that the latter manipulates creative activities. We are not sure that artistic variety in music may be suffered from the standard evaluation of computer music system.

[8]The importance of evaluation by professional musicians becomes more apparent for performance synthesis. Only well-trained musicians can judge the synthesized performance excellent or flawed with their "trained ears".

Evaluation by component comparison The second evaluation method may lead the quantitative evaluation since the comparison requires metrical results which are pervasive scientifically. However this is not necessarily the appropriate evaluation method because some pursue the component technique while some do not. After all, this method can evaluate a system only in one side and may not be legitimate to some types of system.

As for relationships of occurrences, Hiraga [Hir96] and Uwabu [UKI97] proposed their own calculation algorithm: Hiraga's for cognitive understanding, Uwabu's for confirming his model, thus we can quantitatively compare the relationships with them. In this case, a question occurs to which pieces to apply the algorithm. Some may be better in a piece, some in another. Each system chooses its own piece either for their interest or the matching for its purpose. Another question is the meaning of comparison where each system has different purposes. Unless the comparison concerns the purpose of each system, it may make the evaluation meaningless.

5.3.2 Computer music systems and software environment

Object-oriented approach

Software systems have been purposely and intentionally implemented with factors for improving their qualities such as modularity and reusability [Mey88]. Object-oriented approach has been expected to solve problems in the complexity and size of software [Boo95].

A book edited by Pope [Pop91b] introduces computer music systems built with object-oriented approach. Described are computer music systems for orchestra, language (for musical style), composition, sound, and signal processing. They show that, to some extent, music can be described with object-oriented mechanisms of inheritance, encapsulation, and affinity for GUI.

An orchestra system was elegantly written in Smalltalk [Kra91]. It consists of seven classes (Conductor, Orchestra, Player, Instrument, Printed part, Repertoire, and Score) which correspond to actual music activities by human. It looks refined because of the good correspondence of the real world and the design of the system. In MODE (the Musical Object Development Environment) [Pop91a], there are classes to handle sound events. These classes constitute a hierarchy so that it is easy to prepare appropriate repre-

sentation for magnitude of a sound event such as in nominal and numeral.

These two systems represent the two polars of music activities by human. One is the more abstract representation which could be instantiated to indefinite numbers[9]. The other handles absolute values of sound events with the least abstraction in it. We have found no systems for performance synthesis successfully developed by object-oriented approach, neither elsewhere outside of the book.

We adopted object-oriented programming and built classes as in Figure 5.2. The classes in the figure represent those used for performance analysis and synthesis which are shared by systems of Psyche. Though it is possible and we have actually represented the musical structure in the object-oriented mechanism, we are confronted by two defects.

- The hierarchy of musical structure is treated implicitly as an attribute.
- The relationship among occurrences of musical structure along time is also treated implicitly as an attribute which can be shown only relatively where the order of occurrences is available in the system.

The first affects two kinds of musical inheritance based on hierarchy in page 18 (inheritance of performance and analysis items), in spite of the importance of inheritance in performance synthesis. The second affects accompaniment systems of time intensity to work appropriately.

Involving basic information as attributes makes the design and implementation of a system complicated because attributes lack in musical meanings and we can give no priority to them. The design of the system implementation (the design of classes) becomes complicated and not clear in musical meanings.

Object-oriented methods So far computer music systems have claimed to be in the object-oriented approach because they were developed using object-oriented programming languages (OOPL). We mention object-oriented approach by including OOPL and object-oriented methods. Object-oriented methods, by object-oriented analysis (OOA) and object-oriented design (OOD), appeared in 1990s posterior to OOPL. We describe here the applicability

[9]A term "orchestra" could be interpreted in many ways by people. One may think it should be represented with a musical theater, another may with its sponsor.

of object-oriented design to musical information processing, before giving conclusion the applicability of object-oriented approach. Since the design is based on analysis and there are not significant difference in the way of consideration between OOA and OOD, we discuss the applicability of object-oriented design to musical information processing based on a book on OOD [CY91].

OOD is required for the evolution of software systems: evolving in its size, complication, user interface, and domain-oriented system design. The major reason to consider how to construct a system with classes and objects is to

> "efficiently involve information required for problem domain and implementation domain into systems."

This is what we tried to map musical knowledge structure in Figure 2.3 (page 17) to relationships of classes in Figure 5.2 (page 60), that turned to be failed.

OOD handles system complication by *abstraction, encapsulation, inheritance, association, message*, and others. Some of them are realized into OOPL as *class, object, generalization–specialization, whole–part, and attribute–service*. In other words, OOPL can represent subjects which are well described by class, and other implementation methods.

The reasons that some musical information processing systems are well described by OOPL and performance synthesis system is not are as follows.

- Why orchestra is well described?

  Though classes in the system didn't make use of inheritance, each class is independent each other which derives the encapsulation and a simple association among classes.

- Why MODE is well described?

  It manipulates quantitative issues which are good examples of representation by object-oriented method using generalization-specialization method.

- Why performance synthesis cannot be described?

  Because performance synthesis cannot make use of a good characteristic of object-oriented, namely generalization-specialization (a class

hierarchy). Performance synthesis requires representation for recursive hierarchy (page 18), which object-oriented approach does not include.

Relationships among objects of the same class are not supported by object-oriented approach, which leads the inconvenience in representing occurrences along time.

On the other hand, OOD can be useful to design Psyche which includes several systems. We can see the applicability of OOD to Psyche by critical success factors of reusability, readability, performance, and prototyping. Thus OOA and OOD may be valuable for managing Psyche project.

As regards design and implementation of performance synthesis, no necessity is found for OOD to apply performance systems. After all, the most interesting, specialized, and difficult part in the design, namely the description of the problem domain component for performance synthesis could not be solved by OOD.

We can conclude the possibility of object-oriented approach to musical information processing as follows.

applicability
- Apply object-oriented approach to a project as a whole.
- Apply it to a musical component where specialization-generalization relationship exists (making use of class inheritance).
- Apply it to a musical component where each class is independent.

limitation
- Cannot use to represent recursive hierarchy.
- Cannot use the time critical relationship.

Recursive hierarchy is well handled by attribute grammar described below.

Attribute Grammar

As to the inheritance of performance and recursive hierarchy, we used attribute grammar [ASU86] to generate music hierarchy on the parsed tree and annotate it for performance synthesis [HII96]. The detail is described in Appendix. We noticed that attribute grammar is not powerful enough to handle all types of performance parameters, also the semantic rules do not represent musical meaning intuitively.

Possible solutions

Once we actually tried to handle musical meanings into systems, we found the limitation of object-oriented approach and other existing software methodology including attribute grammar. The reasons of the limitation of existing software methodologies are as follows.

- The limitation of representation ability to problem domain with a specialized data structure.
- The complication in manipulating real world activities, especially where temporal meaning is essential.

We want a methodology to smoothly design and implement a system from the problem domain as the quote in page 72. There can be two possible solutions to the problem of the gap between the design and the implementation.

- To prepare by ourselves a software method for music processing systems. The method needs not be general enough such as object-oriented approach, but powerful enough to handle music specific knowledge structure, such as recursive hierarchical structure and time intensity.
- To introduce an intermediate layer between the design of music knowledge and existing software methods. We need a music-specific, namely domain-oriented, or moreover, task-oriented (extremely saying for performance synthesis only) method in this layer.

Though the result is only applicable to musical information processing, the first will be an innovative work which may lead us the general solution of how to generate a domain specific software method. While the second reminds a macro for conventional programming languages, it includes the semantical mapping from the knowledge structure to implementation structure.

5.4 Summary

A computer-assisted musical analysis system Daphne was designed to be used by both professional musicians and computer scientists. It is based on musicology to obtain musical analysis information for performance synthesis

with rules. In order for Daphne to be pervasive, its GUI is realistic with a score and performance which are familiar to musicians.

Because of the small number of implemented musical analysis systems and difficulties in evaluating computer music systems, we haven't evaluated Daphne in the usual evaluation method with subjects and questionaire. An evaluation method applied to many other computer systems such as component comparison by quantification is not necessarily appropriate for computer music systems because of the divergence in the realization of a system.

We found the applicability and limitation of existing software method, object-oriented approach, to musical processing systems. The limitation of object-oriented approach can be fatal to some computer music systems.

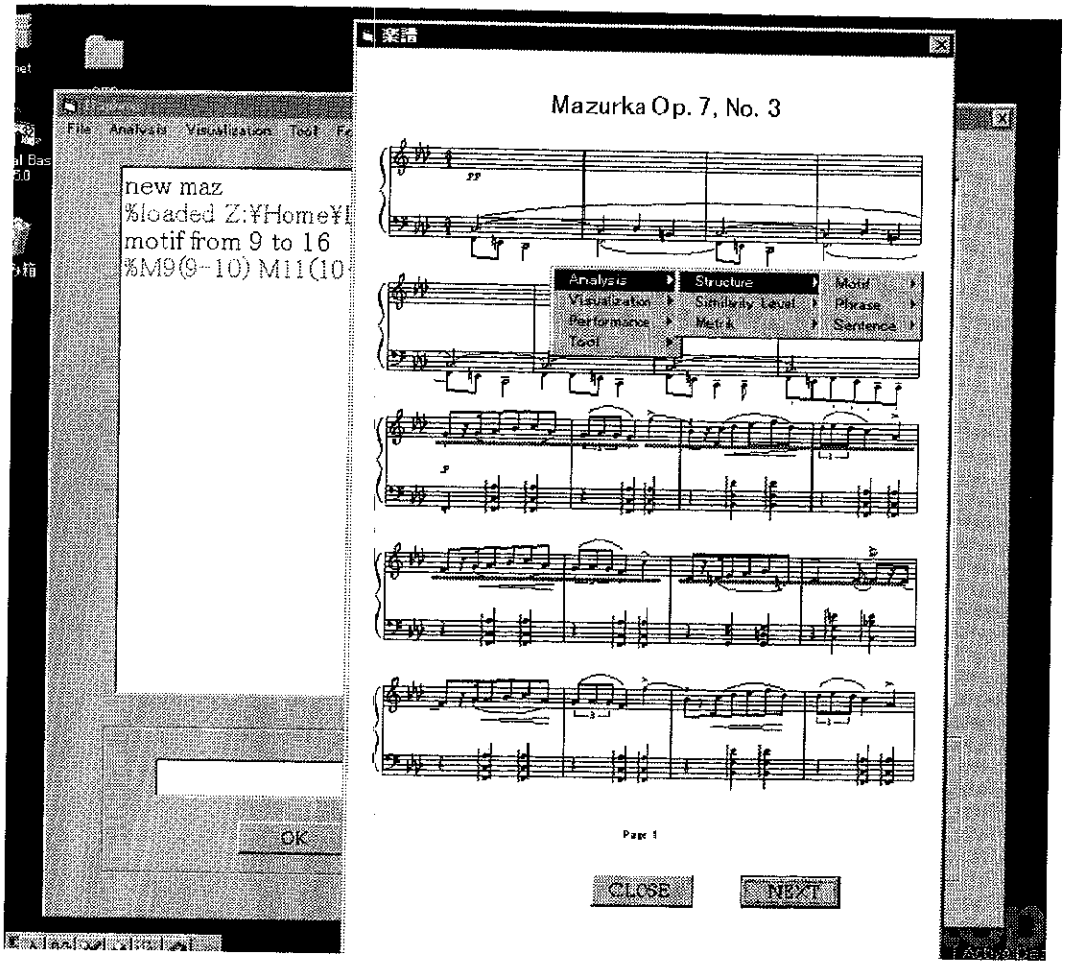
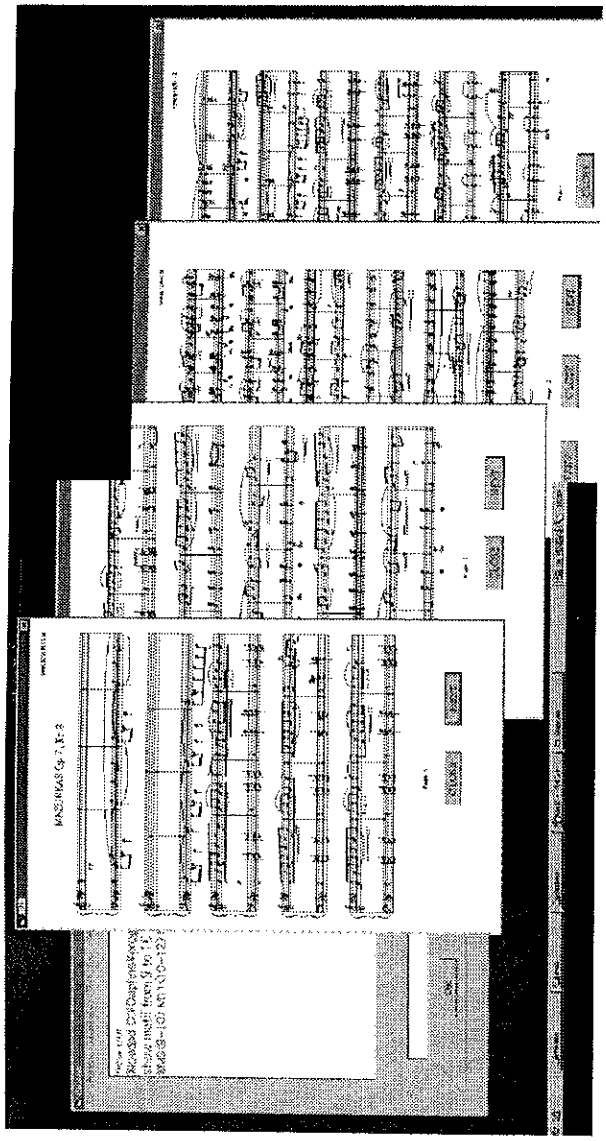


Figure 5.6: GUI of Daphne (2): Analysis by WYSIWYG



Using two monitors. Because of the difference of resolution in the monitors, display image is cut in some part.

Figure 5.7: GUI of Daphne (3): Analysis by WYSIWYG (many score sheets)

Chapter 6

Conclusion

6.1 Contributions

We have described the significant role of musical structure to musical information processing especially for performance synthesis and shown computer music systems that are valuable both for musicology and computer music research in performance analysis and synthesis.

A performance visualization system based on musical structure can be regarded as a new generation sequence software which visualizes performance with the following novelties.

- Performance rationale can be understood with the structure based visualization.
- Relationships of performance reflecting musical structure can be understood at ease.
- Comparison of performances can lead performance rules.
- Functions for performance synthesis enable users to edit performance with the visual assistance.
- Functions of constraining and pervading performance editing assist users in modification of performance by keeping the consistency and musical artistry.

The visualization system was evaluated to be useful for the analysis and synthesis of performance while some requirements on usability and editing functions were commented.

A computer-assisted musical analysis system Daphne for obtaining and storing musical analysis information was designed and implemented. Analyzed information is used for performance synthesis. Daphne provides WYSIWYG user interface for analysis with displayed musical score and performance sound in order for professional musicians to use it. Daphne eliminates ambiguity of musical analysis so that users and systems can share analysis information.

Through the design and implementation of Daphne, the following became clear.

- Structure of musical knowledge, where musical structure is the primary with attributive analysis information.
- The applicability and limitation of object-oriented approach to music information processing, especially for performance synthesis.

  The innate natures of hierarchy of musical structure, the inheritance of performance for occurrences in the upper layer, and time intensity of performance are not represented by object-oriented approach.

Since it is difficult to evaluate and compare computer music systems for their divergence in purpose, leaning theory, and the way a system is used, we have to be careful in evaluating Daphne. We also discussed the difficulties in evaluating computer music systems.

6.2 Future Research Issues

In order to establish a method to synthesize expressive performance with rules, we are going to enhance Daphne and manipulation of a performance rule set and its engine. Some of the manipulations are handled by Leda (Logical Expression of Dynamics and Agogics).

Issues on Daphne include the implementation of analysis for harmony and chord progression and for tonality involved in Daphne's WYSIWYG GUI. The evaluation of Daphne, especially by professional musicians, is required to make it pervasive as a computer-assisted musical analysis system among

the wide range of users. Also the evaluation of Daphne should give a guide to evaluation method of computer music systems.

Other issues to be clarified regarding performance synthesis are as follows.

- Formalization of rules.

  Analysis information obtained through Daphne is an object to which a performance rule is applied. The formalization clarifies how to use Daphne's information in rules.

  We expect that the structure of rules and global environment become clear by the formalization. For example, meta rule and necessary parameters commonly used by rules may be found to improve the rule system.

- Quantification of performance parameters.

  In order for performance synthesis systems to be artificially intelligent, quantification of performance parameters should be automated to some extent, though not the complete automation is required for expressive performance.

- Dependency and independency of performance parameter.

  Human players do not take agogics and dynamics separately in playing, while performance rules tend to treat them as performance parameters independently. The dependency among parameters is difficult to be modeled in computer music systems.

- Steps to complete the whole music.

  Human players generate their music by processing hands and feet concurrently, not by melody then accompanied harmony. Also a section of a musical piece is not generated independently from other divisions. These concurrent processings both in time and (score) space should be in concern in performance synthesis by systems.

Computer music research has many research issues which will be solved by musical structure.

Acknowledgements

I owe a deep debt of gratitude and appreciation to Professor Shigeru Igarashi, my thesis advisor for all his guidance, support, and enthusiasm. I would like to express my appreciation to other committee members, Professor Tetsuo Ida, Professor Keinosuke Nagai, Professor Jiro Tanaka, and Professor Chiharu Hosono for their intellectual advices.

I would like to acknowledge Prof. Tetsuya Mizutani and Prof. Masayuki Shio for their considerate help and suggestions in matters both technical and editorial.

Thanks to all the students of Psyche project for their challenge to the pioneering research. In particular, I would like to thank Jian-Li Liu, who implemented Daphne musical analysis system.

Finally, I would like to appreciate my family, Yuzuru and Syun, for their constant encouragement.

Bibliography

[AKI97] Y. Aono, H. Katayose, and S. Inokuchi. Extraction of expression parameters with multiple regression analysis. *Journal of Information Processing*, 38(7):1473–1481, 1997.

[ASU86] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers - Principles, Techniques, and Tools.* Addison-Wesley, 1986.

[Bal92] M. Balaban. Music structures: Interleaving the temporal and hierarchical aspects in music. In Balaban, Ebcioglu, and Laske, editors, *Understanding Music with AI*, pages 110–139. The MIT Press, 1992.

[Bal96] M. Balaban. The music structures approach to knowledge representation for music processing. *Computer Music Journal*, 20(2):96–111, 1996.

[BDCM93] K. Barbar, M. Desainte-Catherine, and A. Miniussi. The semantics of musical hierarchies. *Computer Music Journal*, 17(4):30–37, 1993.

[BDPV92] R. Bresin, G. De Poli, and A. Vidolin. Symbolic and sub-sybolic rules system for real time score performance. In *Proc. of ICMC*, pages 211–218. ICMA, 1992.

[Ber87] W. T. Berry. *Structural Functions in Music.* Dover Publications, 1987.

[Boo95] G. Booch. *Object-oriented Analysis and Design with Applications—second edition.* Addison-Wesley, 1995.

[BWE94] S. Brewster, P. Wright, and A. Edwards. The design and evaluation of an auditory-enhanced scrollbar. In *Proc. of CHI '94 Conference*, pages 173–179. ACM SIGCHI, 1994.

[Cen] Center for Computer Assisted Research in the Humanities. *The Humdrum Manual.*

[CH87] M. Critchley and R. A. Henson, editors. *Music and the Brain.* Science Publisher, 1987.

[Cha97] C. Chafe. Statistical pattern recognition for prediction of solo piano. In *Proc. of ICMC*, pages 145–148. ICMA, 1997.

[CM60] G. Cooper and L. B. Meyer. *The Rhythmic Structure of Music.* University of Chicago Press, 1960.

[CO96] C. Chafe and S. O'Modhrain. Musical muscle memory and the haptic display of performance nuance. In *Proc. of ICMC*, pages 428–431. ICMA, 1996.

[Com] Come On Music. *Recomposer ver2.5f.*

[CY91] P. Coad and E. Yourdon. *Object-Oriented Design.* Prentice-Hall, 1991.

[Dan93] R. B. Dannenberg. Music representation issues, techniques, and systems. *Computer Music Journal*, 17(3):20–30, 1993.

[Deg96] B. Degazio. A computer-based editor for Lerdahl and Jackendoff's rhythmic structure. In *Proc. of ICMC*, pages 396–397. ICMA, 1996.

[Deu82] D. Deutsch, editor. *The Psychology of Music.* The Academic Press, 1982.

[Deu87] D. Deutsch. *Music and the Brain*, chapter 7, pages 131–170. Science Publisher, 1987.

[FKI90] T. Fukuoka, H. Katayose, and S. Inokuchi. Music synthesis stage in music interpretation system. In *Proc. of the Annual Conference of IPSJ*, number 40, pages 1585–1586. IPSJ, 1990.

[Fri91] A. Friberg. Generative rules for music performance: A formal description of a rule system. *Computer Music Journal*, 15(2):56–71, 1991.

[Fuj95] Y. Fujiwara. *The natural performance of rhythm.* The Hakusui Press, 1995.

[Hat87] G. Hatano, editor. *Music Cognition.* University of Tokyo Press, 1987.

[HCW85] P. Howell, I. Cross, and R. West, editors. *Musical Structure and Cognition.* The Academic Press, 1985.

[HHF96] T. Hoshishiba, S. Horiguchi, and I. Fujinaga. Study of expression and individuality in music performance using normative data derived from MIDI recordings of piano music. In *Proc. of ICMPC*, pages 465–470, 1996.

[HI97] R. Hiraga and S. Igarashi. Psyche: University of Tsukuba, computer music project. In *Proc. of ICMC*, pages 297–300. ICMA, 1997.

[HII96] R. Hiraga, S. Igarashi, and A. Iyatomi. Attribute grammar for music processing system. In *Proc. of Annual Conference of Japan Society for Software Science and Technology*, number 13, pages 245–248. JSSST, 1996.

[HIL97] R. Hiraga, S. Igarashi, and J. L. Liu. An interactive music interpretation system Daphne. In *Proc. of 1997 Japan-China Joint Meeting on Musical Acoustics*, pages 99–102, 1997.

[HIM96a] R. Hiraga, S. Igarashi, and Y. Matsuura. Music processing and object-orientedness. In *Proc. of Programming Symposium*, number 37, pages 207–214. IPSJ, 1996.

[HIM96b] R. Hiraga, S. Igarashi, and Y. Matsuura. Visualized music expression in an object-oriented environment. In *Proc. of ICMC*, pages 483–486. ICMA, 1996.

[HIM97a] R. Hiraga, S. Igarashi, and Y. Matsuura. An integrated musical performance visualization system. *Journal of Information Processing*, 38(11):2391–2397, 1997.

[HIM97b] R. Hiraga, S. Igarashi, and Y. Matsuura. A musical performance visualization system with object-oriented approach. *Computer Software*, 14(1):50–43, 1997.

[Hir96] Y. Hiraga. A cognitive model of pattern matching in music. In *Proc. of ICMC*, pages 248–250. ICMA, 1996.

[Hir97a] R. Hiraga. Objective evaluation for computer generated musical performance. In *Workshop on Issues in AI and Music - Evaluation and Assessment*, pages 23–26. International Joint Conference on Artificial Intelligence, 1997.

[Hir97b] K. Hirata. A report on IJCAI '97 workshop on Issues in AI and Music—Evaluation and Assessment. In *SIGMUS*, number 22, pages 19–24. IPSJ, 1997.

[Hir97c] K. Hirata, editor. *Workshop on Issues in AI and Music - Evaluation and Assessment.* International Joint Conference on Artificial Intelligence, 1997.

[Hir98a] R. Hiraga. Synthesizing expressive performance. In Y. Nagashima, S. Hashimoto, Y. Hiraga, and K. Hirata, editors, *bit Computer and Music (II)*, pages 270–282. Kyoritsu Shuppan, 1998.

[Hir98b] K. Hirata. Representation of musical knowledge on computer. In Y. Nagashima, S. Hashimoto, Y. Hiraga, and K. Hirata, editors, *bit Computer and Music (II)*, pages 163–181. Kyoritsu Shuppan, 1998.

[HLI99] R. Hiraga, J. L. Liu, and S. Igarashi. A computer-assisted music analysis system for sharing music knowledge. *Japan Society of Artificial Intelligence*, 14(3), 1999.

[Hon92] H. Honing. Expresso, a strong and small editor for expression. In *Proc. of ICMC*, pages 215–218. ICMA, 1992.

[Hur] D. Huron. Humdrum — frequently asked questions. URL: http://www.lib.virginia.edu/dmmc/Music/Humdrum/humdrumFAQ.html.

[HWC91] P. Howell, R. West, and I. Cross, editors. *Representing Musical Structure*. The Academic Press, 1991.

[I+94] S. Inokuchi et al. *Kansei Information Processing*. Ohm-sha, 1994.

[IHI97] A. Iyatomi, R. Hiraga, and S. Igarashi. Unfolding performance expression with rules based on music structure. In *Proc. of the Annual Conference of JSAI*, number 11, pages 280–282. JSAI, 1997.

[IHIM96] S. Igarashi, R. Hiraga, A. Iyatomi, and Y. Matsuura. Visualization of music interpretation based on its structure. In *Proc. of ICCMMS*, pages 21–26, 1996.

[II95] A. Iyatomi and S. Igarashi. Making computerized piano performance artistic reflecting musical structures. In *Proc. of the Annual Conference of JSAI*, number 9, pages 621–622. JSAI, 1995.

[IOD+96] S. Igarashi, D. Ogawa, G. Dai, A. Iyatomi, and Y. Matsuura. Analyses and applications of expressive musical performance using phrase structure. In *Proc. of the Annual Conference of IPSJ*, number 52, pages 1-441–442. IPSJ, 1996.

[ITC+95] S. Igarashi, T. Tsuji, D. Chiba, M. Matsushita, D. Ogawa, A. Iyatomi, and K. Seino. Representation of expressions in music performances and its applications to accompaniment systems. In *Proc. of Programming Symposium*, number 36, pages 47–56. IPSJ, 1995.

[ITMH93] S. Igarashi, T. Tsuji, T. Mizutani, and T. Haraguchi. Experiments on computerized piano accompaniment. In *Proc. of ICMC*, pages 415–417. ICMA, 1993.

[Kaw98] Kawai. *ScoreMaker 2.0*, 1998.

[Kel87] H. Keller. *Phrasierung und Artikulation.* Ongaku-no-tomo-sha, 1987.

[Ken98] M. Kennedy. The Oxoford dictionary of music—second edition, 1998.

[KFI90] H. Katayose, T. Fukuoka, and S. Inokuchi. Performance-rule extraction in music interpretation system. In *Proc. of the Annual Conference of IPSJ*, number 40, pages 1587–1588. IPSJ, 1990.

[KIIH98] H. Koike, M. Ichiyanagi, S. Igarashi, and R. Hiraga. Analysis by synthesis of expressive performance with agogics. In *Proc. of the Annual Conference of JSAI*, number 12, pages 520–521. JSAI, 1998.

[Kor95] A. Kornstadt. Score-to-Humdrum: A graphical environment for musicological analysis. In *Computing in Musicology, vol. 10*, pages 105–122. Center for Computer Assisted Research in the Humanities, 1995.

[Kra91] G. Krasner. Machine tongues VIII: The design of a Smalltalk music system. In S. T. Pope, editor, *The Well-Tempered Object*, pages 7–17. The MIT Press, 1991.

[KT94] H. Katayose and Y. Takeuchi. Music theories on interpretation and its application to music. In *SIGMUS*, number 7, pages 15–22. IPSJ, 1994.

[Kum95] T. Kumata. *Music Analysis for Performance.* Ongaku-no-tomo-sha, 1995.

[LJ83] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music.* The MIT Press, 1983.

[LO94] J. Larue and M. Ohmiya. *Methods and Models for Comprehensive Style Analysis.* Ongaku-no-tomo-sha, 1994.

[Mey56] L. B. Meyer. *Emotion and Meaning in Music.* University of Chicago Press, 1956.

[Mey88] B. Meyer. *Object-Oriented Software Construction.* Prentice Hall, 1988.

[MI95] M. Matsushita and S. Igarashi. Computerized music performance system cooperating with intermittent play by persons. In *Proc. of the Annual Conference of JSAI*, number 9, pages 623–626. JSAI, 1995.

[MKKN96] T. Murakami, T. Kumagai, Y. Kajikawa, and Y. Nomura. Extraction of a performer's characteristic on tempo by neural network. In *Proc. of the Annual Conference of IPSJ*, number 52, pages 1–437–438. IPSJ, 1996.

[Mor] S. Moroi. The commentary of Beethoven's piano Sonaten. zen-on piano library, Beethoven, Sonaten 2.

[Mur87] H. Murao. Cognition in musical analysis. In G. Hatano, editor, *Music Cognition*, pages 1–40. University of Tokyo Press, 1987.

[MZ94] G. Mazzola and O. Zahorka. The Rubato performance workstation on NEXTSTEP. In *Proc. of ICMC*, pages 102–108. ICMA, 1994.

[MZN95] G. Mazzola, O. Zahorka, and T. Noll. Musical analysis on the Rubato platform. In *Computing in Musicology, vol. 10*, pages 143–149. Center for Computer Assisted Research in the Humanities, 1995.

[Nar92] E. Narmour. *The Analysis and Cognition of Basic Melodic Structures.* University of Chicago Press, 1992.

[NHHH98] Y. Nagashima, S. Hashimoto, Y. Hiraga, and K. Hirata, editors. *bit Computer and Music (II).* Kyoritsu Shuppan, 1998.

[NS87] H. Nishimura and SIGMUS, editors. *bit Computer and Music.* Kyoritsu Shuppan, 1987.

[Ock91] A. Ockelford. The role of repetition in perceived musical structures. In P. Howell, R. West, and I. Cross, editors, *Representing Musical Structure*, pages 129–160. The Academic Press, 1991.

[ODI96] D. Ogawa, G. Dai, and S. Igarashi. Computerized piano accompaniment. In *SIGMUS*, number 14. IPSJ, 1996.

[OI95] H. Ono and S. Igarashi. Analyses of percussion performances with applications to person-machine ensemble systems. In *Proc. of the Annual Conference of IPSJ*, number 50, pages 1–357–358. IPSJ, 1995.

[Opp92] D. Oppenheim. Compositional tools for adding expression to music. In *Proc. of ICMC*, pages 223–226. ICMA, 1992.

[Osa98] N. Osaka. Computer music and sound synthesis technique. In Y. Nagashima, S. Hashimoto, Y. Hiraga, and K. Hirata, editors, *bit Computer and Music (II)*, pages 41–54. Kyoritsu Shuppan, 1998.

[OW96] D. Oppenheim and J. Wright. Towards a framework for handling musical expression. In *Proc. of ICMC*, pages 71–74. ICMA, 1996.

[Poo95] O. E. Pool. The Apollo project: Software for musical analysis using Darms. In *Computing in Musicology, vol. 10*, pages 123–130. Center for Computer Assisted Research in the Humanities, 1995.

[Pop91a] S. T. Pope. Introduction to MODE: The musical object development environment. In S. T. Pope, editor, *The Well-Tempered Object*, pages 83–106. The MIT Press, 1991.

[Pop91b] S. T. Pope, editor. *The Well-Tempered Object.* The MIT Press, 1991.

[Ret95] R. Reti. *The Thematic Process in Music.* Ongaku-no-tomo-sha, 1995.

[Rie84] H. Riemann. *Musikalische Dynamik und Agogik — Lehrbuch der musikalischen Phrasierung.* Breitkopf und Hartel, 1884.

[Roa96] C. Roads, editor. *The Computer Music Tutorial.* The MIT Press, 1996.

[Slo82] J. A. Sloboda. Music performance. In D. Deutsch, editor, *The Psychology of Music*, pages 479–496. The Academic Press, 1982.

[SMI84] R. Sasakawa, K. Miyoshi, and S. Igarashi. The design and implementation of a generic music description language. In *Proc. of Programming Symposium*, number 25, pages 16–25. IPSJ, 1984.

[STT97] T. Suzuki, K. Tokunaga, and H. Tanaka. Example based approach for production of expressive performance. In *SIGMUS*, number 21, pages 7–12. IPSJ, 1997.

[Tag87] T. Taguti. Automated musical performance and its model. In Nishimura and SIGMUS, editors, *bit Computer and Music*, pages 42–53. Kyoritsu Shuppan, 1987.

[Tau94] D . Taupin. *Music TeX*, 1994.

[TS95] W. F. Thompson and M. Stainton. Using Humdrum to analyze melodic structure: An assessment of Narmour's Implication-Realization model. In *Computing in Musicology, vol. 10*, pages 24–33. Center for Computer Assisted Research in the Humanities, 1995.

[UKI97] Y. Uwabu, H. Katayose, and S. Inokuchi. A structural analysis tool for expressive performance. In *Proc. of ICMC*, pages 121–124. ICMA, 1997.

[VSK89] H. H. Vogt, S. D. Swierstra, and M. F. Kuiper. Higher order attribute grammars. In *Proc. of ACM SIGPLAN '89, Conference on Programming Language Design and Implementation*, pages 131–145. ACM, 1989.

[Wal87] A. Walker. *A Study in Musical Analysis.* Ongaku-no-tomo-sha, 1987.

[Wid93] G. Widmer. Understanding and learning musical expression. In *Proc. of ICMC*, pages 268–275. ICMA, 1993.

[Wid94] G. Widmer. Learning expression at multiple structural levels. In *Proc. of ICMC*, pages 95–101. ICMA, 1994.

[WMSH93] G. Wiggins, E. Miranda, A. Smaill, and M. Harris. A framework for the evaluation of music representation systems. *Computer Music Journal*, 17(3):31–42, 1993.

[Yag97] A. Yagi. Analysis and justification of piano performance through MIDI. Undergraduate Thesis, Univ. of Tsukuba, 1997.

[Yam98] Yamaha. *XGWorks 2.0*, 1998.

Appendix
Performance Synthesis by Attribute Grammar

In order to generate expressive performance data, values of performance parameters on tempo and dynamics have to be passed to rules. Higher order Attribute Grammar (HAG) [VSK89] was applied to the design of Daphne in order to obtain performance parameter values based on a musical interpretation model. The model uses relationships between occurrences of musical structure.

Attribute Grammar (AG) is thought to be usable in the manipulation of musical structure, since occurrences of music structure forms a tree, although the first order AG's semantic rules are attached to a parse tree which is intuitively different from the musical structure tree (MST). Another limitation of the first order AG is manipulation of several expressions. It is often the case that more than one expressions of Daphne which declare parental (hierarchical) relationships among occurrences of musical structure make an MST of a piece of composition. Thus the concept of non terminal attribute (NTA) of HAG is applicable in order to solve these two issues.

Using the sample score from Chopin's Mazurka Op. 7, No. 3 in Figure A.1, whose musical structure is shown in Figure A.2, and performance rules below, we will show how HAG is applied to synthesize performance.

R 3 *when a consecutive occurrence of musical structure is considered almost the same as the previously appearing occurrence (we call such occurrence a seed occurrence), then the latter occurrence (a referring occurrence) is played softer than the first one (the same as R2 in page 28).*

R 4 *when a consecutive occurrence of musical structure is considered almost the same while its pitch is higher than the seed occurrence, then the referring occurrence is played louder than the seed.*

R 5 *when a consecutive occurrence of musical structure is considered almost the same while its pitch is lower than the seed occurrence, then the referring occurrence is played softer than the seed.*

R 6 *when half cadence appears, then the tempo is getting slower.*

R 7 *when an occurrence is in the middle of the larger musical structure, the tempo is faster than occurrences at the beginning and the close of the larger structure.*

By applying the above rules to data of musical structure, the interpretation is acquired as in Figure A.3.

Musical analysis obtained through Daphne is stored in three ways (page 59). Using objects for analysis, we can construct analysis representation by language. Figure A.4 shows how the analysis of the example score is written in a musical analysis language. Figure A.5 shows the part of BNF notation.

Attribute values attached to nodes and leaves of an MST are as follows. The subscript denotes the level of unit. Unit level 3 in τ and ρ stands for motif. In other words, these attributes have meaningful value only in motifs. The smaller the value of j, the larger the unit (2 for phrase and 1 for sentence).

- *synthesized attribute values.*
  - υ_j : The total amount of change in volume regarding musical structure. $\upsilon_j = \sigma_j + \iota_j$, where σ and ι are inherited attributes described below.
  - τ_3 : The different value in tempo of a motif.
  - ρ_3 : The rate of changing tempo derived from cadence or half cadence.
- *inherited attribute values.*
  - ι_j : The total amount of change in volume in occurrences of the larger musical structure unit. $\iota_j = \sum_{k=0}^{j-1} \sigma_k$

– σ_j : The different value obtained by a relationship of sibling occurrences. Say two occurrences are i_1 and i_2 where the i_1 is a seed occurrence,

$$\sigma_j(i_1) = 0$$

$$\sigma_j(i_2) = \mathcal{V}(\mathcal{V}(i_1), j, rel)$$

Here $\mathcal{V}$(unit_occurrence) gives volume values from unit_occurrence. The other $\mathcal{V}$, which has three arguments, calculates the different volume value from the level of unit in its second and the relationship (ostinato, for example) between occurrences in its third arguments.

Each node is represented as a record of {symbol, Att, NT}, where Att is a set of attributes. In this case, Att=$\{\upsilon, \rho, \tau, \iota, \sigma\}$. NT is a nonterminal attribute which is valid only when a node is a leaf. The valid value of NT is either NULL or a subtree whose root is the node.

Attribute values in leaves are used to obtain the concrete performance data. For other nodes which have child nodes, the concatenation of musical interpretation in child nodes represented by attribute values are regarded to be the interpretations. Notice that some leaves are nonterminal attribute (NTA) which are annotated at the parsing of an expression by adopting HAG.

Information of musical structure is declared in any order. There is no restriction in the syntax to regulate the order of declarations. Therefore, hierarchical relationships declared in Figure A.4, namely (9) and (10), may appear reversely. Synthesized attributes ρ and τ have no dependency on any other attribute. On the other hand, σ for instances of a higher level unit is desired to be obtained prior to their child nodes.

Higher order Attribute Grammar (HAG) is more usable than the first order attribute grammar in the manipulation of MST. Nonterminal Attribute (NTA) of HAG is used both for obtaining an MST for an expression and for constructing a complete MST from several expressions. The annotation of the complete MST is required to give all attribute values to all nodes of the MST.

How to obtain an MST is shown first. Figure A.5 shows the part of BNF notation of the current Daphne. The subtrees of the MST in the sample score (Figure A.2) are obtained one by one from expression. For example,

expression (9) in Figure A.4 is parsed with the definition of the latter half of (D6). For obtaining a subtree which corresponds to the expression (9), a nonterminal attribute (NTA) is introduced to the production rule. The production rule for the latter half of (D6) is redefined as follows.

$$U \rightarrow E_1 \overline{E_2}$$

E_1 represents the latter half of (D6), while $\overline{E_2}$ is an NTA which is evaluated to be an MST. The MST has a node *par1* as a parent and leaves *stc1* and *stc2* as children. Moreover these two leaves are also NTAs which are annotated to have subtrees for the parental relationship of sentences and motifs.

Figure A.6 shows production rules and semantic rules regarding the construction of an MST from the latter half of (D6). Semantic rules of E1 is devoted to generate an MST. In this way, several MSTs are built from several expressions which represent a hierarchical relationship. After parsing all expressions, several attribute values are obtained. Then the complete MST is annotated in the following order.

1. Joining an MST into a leaf of another tree if the leaf is a root of another MST to make the complete MST.

2. The complete MST is searched in the top down order to get synthesized values v.

Only the outline of applying HAG to the processing of musical structure is described here. HAG is usable to some extent in calculating attribute values for MST.

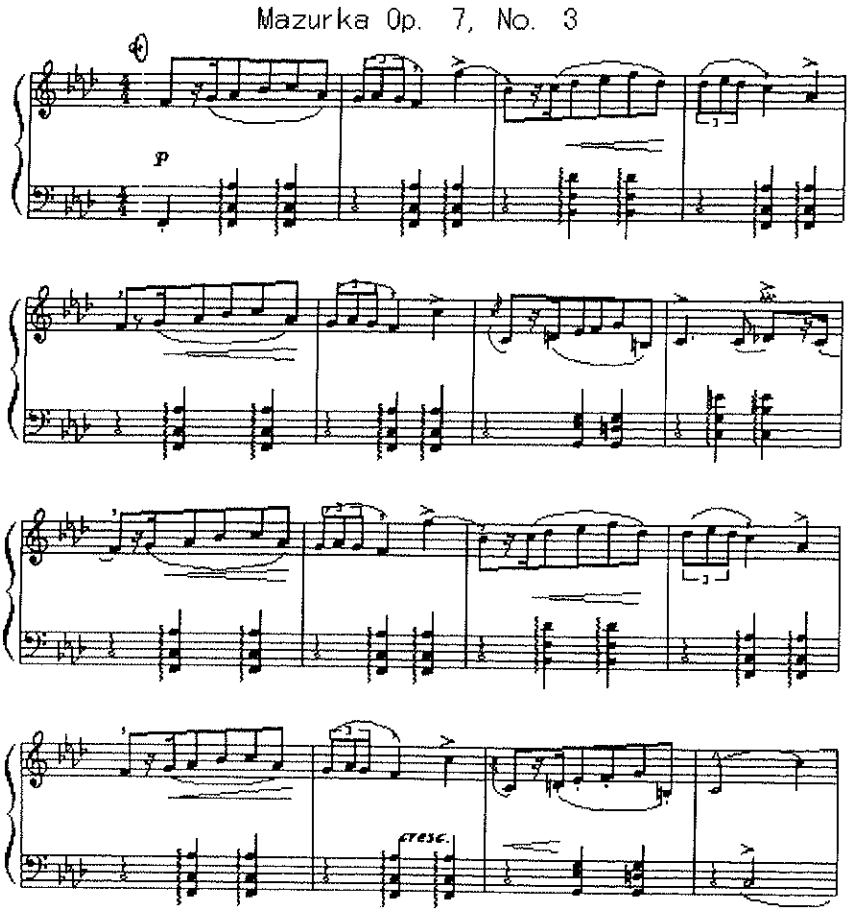


Figure A.1: Mazurka Op. 7, No. 3, by Chopin (both hands) —ScoreMaker

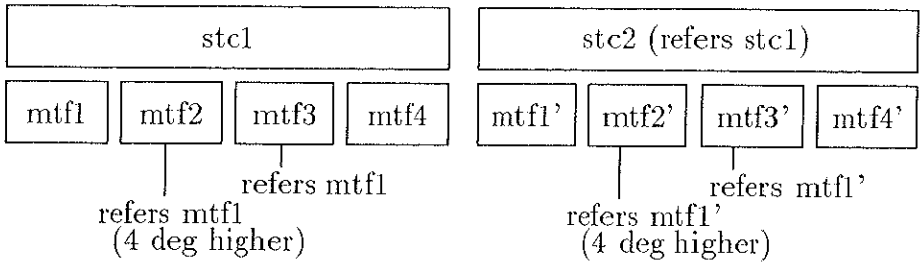


Figure A.2: Musical structure (hierarchical structure and relationships between occurrences)

$$\mathcal{I}_v(par1) = \mathcal{I}_v(stc1); \mathcal{I}_v(stc1) - \Delta^{ost}_{stc}. \quad \text{(I)}$$

$$\mathcal{I}_v(stc1) = \mathcal{I}_v(mtf1); \mathcal{I}_v(mtf1) + \Delta^{up}_{mtf}; \mathcal{I}_v(mtf1) - \Delta^{ost}_{mtf}; \mathcal{I}_v(mtf4) \quad \text{(II)}$$

$$\mathcal{I}_t(par1) = \mathcal{I}_t(stc1); \mathcal{I}_t(stc1) + \Delta^{hcadence}_{stc}. \quad \text{(III)}$$

$$\mathcal{I}_t(stc1) = \mathcal{I}_t(mtf1); \mathcal{I}_t(mtf1) - \Delta^{mid}_{stc}; \mathcal{I}_t(mtf1) - \Delta^{mid}_{stc}; \mathcal{I}_t(mtf4) \quad \text{(IV)}$$

remarks:

- $\mathcal{I}_v(par1)$: interpretation regarding dynamics (volume) for the sixteen measures.
- $\mathcal{I}_t(stc1)$: interpretation regarding tempo for the first sentence (eight measures).
- $+\Delta^{ost}_{stc}$: volume (tempo) is up (down) by Δ because of *ostinato sentence*.

Figure A.3: Interpretation of the sample score

piece(CM7-3-9-24, b3/4, <(par1))	(1)	par(par1, <(stc1, stc2))	(9)
par(par1, bp9.1, ep24.3)	(2)	stc(stc1, <(mtf1, mtf2, mtf3, mtf4))	(10)
stc(stc1, bp9.1, ep16.3)	(3)	seq(mtf2, mtf1)	(11)
stc(stc2, bp17.1, ep24.3)	(4)	higher(mtf2, mtf1, 4)	(12)
mtf(mtf1, bp9.1, ep10.2)	(5)	ost(mtf3, mtf1)	(13)
mtf(mtf2, bp10.3, ep12.3)	(6)	ost(stc2, stc1)	(14)
mtf(mtf3, bp13.1, ep14.3)	(7)	half_cadence(stc2)	(15)
mtf(mtf4, bp15.1, ep16.3)	(8)		

Figure A.4: Description of musical structure in Daphne

(D1) $< daphne-decl >::= piece(< name >, < beat >, < unit-seq >) < decls >$

(D2) $< decls >::=< unit-decl > | < rel-decl > | < unit-attrib >$

(D3) $< unit-seq >::=< (< unit-names >)$

(D4) $< unit-names >::=< unit-name > | < unit-name >< unit-names >$

(D5) $< unit-name >::=< name >$

(D6) $< unit-decl >::=< unit > (< unit-name >, < begin-place >, < end-place >) | < unit > (< unit-name >, < unit-seq >)$

(D7) $< unit >::= par|stc|ph|mtf|seq|psg$

(D8) $< rel-decl >::= seq(< unit-name >, < unit-name >) < hl > |ost(< unit-name >, < unit-name >)$

(D9) $< hl >::= higher(< unit-name >, < unit-name >)|lower(< unit-name >, < unit-name >)$

(D10) $< unit-attrib >::= cadence(< unit-name >)$

Figure A.5: BNF of an analysis language (part)

Production Rule	Semantic Rule
$U \rightarrow E_1 E_2$	$E_2 := E_1.mst$
$E_1 \rightarrow E$	$E_1.mst := E.mst$
$E \rightarrow T(P, < C >)$	$E.mst := build\_tree(P, C)$
$T \rightarrow unit\_ident$	
$P \rightarrow unit\_name$	$P.sym := unit\_name$
$C \rightarrow S, C$	$C_0.sym := cat(sym(S), sym(C_1))$
$S \rightarrow unit\_name$	$S.sym := unit\_name$
$C \rightarrow S$	$C.sym := S.sym$

Functions:
build_tree(P, C) : make a tree rooted by P and C's components as children of P.
sym(N) : returns a set of symbols of components of N.

Figure A.6: Rules to obtain an MST from an expression