楽曲に対する多様な解釈を扱う音楽アノテーションシステム

梶 克彦[†] 長尾 確[††]

一般にテキスト，ビデオ，音楽などのコンテンツには複数の解釈が存在し，それらのコンテンツを対象とする場合，複数の解釈の中から適切なものを扱う必要がある．そこで我々は任意のコンテンツに対する複数の解釈を扱うためのプラットフォームとして Annphony を構築した．Annphony では，一般的なアノテーション記述形式である RDF を拡張し，全てのアノテーションに識別子を付与することでそれぞれの解釈を区別して扱うことができる．解釈が多様になりやすいコンテンツの典型例として音楽が挙げられる．一般に音楽は必ずしも解釈を一意に決定できず，その楽曲を捉える人によって異なる解釈を見出すことがしばしばみられる．そこで我々は Annphony を基盤とし，楽曲に対する多様な解釈を Web 上のユーザから取得する音楽アノテーションシステムを構築した．本システムは 1. 書誌情報などの楽曲自体に対するアノテーション，2. 連続メディアに対するアノテーション，3. 楽譜に対するアノテーションの 3 種類のエディタを備える．また実際に本システムによりアノテーションを収集し，楽曲に対する多様な解釈を扱う例として，楽曲検索システムとプレイリスト作成支援システムを構築した．

A Music Annotation System for Multiple Interpretations of Tunes

Katsuhiko Kaji[†] and Katashi Nagao [††]

Generally, there are multiple interpretations of single content such as a text, a video clip, a song, and so on. Therefore we have developed an annotation platform called “Annphony” to deal with such multiple interpretations. By extending RDF, a common annotation framework, each annotation has its own identifier that enables distinction of each interpretation. Music is one typical example that apt to cause multiple interpretations. Listeners do not necessarily make the same interpretation about the tune. Therefore, to deal with various musical interpretations we have developed a musical annotation system that consists of the following annotation editors - 1: basic information of the tunes , 2: continuous media, and 3: musical scores. We have also developed two applications based on annotations about multiple interpretations collected by the annotation system. One is a music retrieval system based on music’s inner structure. The other is a playlist recommendation system that semi-automatically generates playlists that reflect listener preferences and situations.

1. はじめに

近年，インターネット上に氾濫する Web ページやマルチメディアコンテンツを，単に視聴するだけでなく高度に利用したいという欲求が高まっている．しかし，コンテンツを自動的に解析して意味を推論し，知的な処理を行うには明らかに技術的な限界がある．一般に人間の作成するコンテンツには多義性と文脈依存性が内在しているため，人間自身による解釈が伴わなければコンテンツの意味を捉えることができない．そこで人間の解釈を機械的な処理に適用可能な形式で取り込むために，アノテーションに関する研究[1)2)3)4)5)]が進められている．しかしこれらの研究にはいくつかの解決すべき問題が含まれている．

コンテンツには複数の解釈が存在し，それらの中からコンテンツを利用する人にとって適切な解釈を推論する必要がある．言語学における語用論の分野では，ある文を意味的に捉えるために，その文法や辞書的な意味に加えて，発話した人や背景を踏まえる必要があるといわれている．発話した人や背景によっては，一文に複数の解釈が存在し，一意に決定することが困難な場合が存在する．このような文の意味を考慮するためには，考えうる解釈の中から，様々な文脈や知識の関連性を踏まえて適切なものを推論する処理が必要となる．しかし従来のアノテーションに関する研究[1)2)3)4)5)] は，それぞれのコンテンツに対して，唯一の事実をユーザから取得し扱うための枠組みであり，複数の解釈を扱うことには適していない．

コンテンツに対する解釈を幅広く収集するためのシステムは，Web システムとして提供され，また収集さ

† 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya Univ.
†† 名古屋大学情報メディア教育センター
Center for Information Media Studies, Nagoya Univ.

れたアノテーションは Web 上で共有されることが望ましい．一般にアノテーションを取得するための人的コストが非常に高いことが問題とされている[6]が，その解決策の一つとして Folksonomy[7] が挙げられる．del.icio.us[8] や Flickr[9] において，Web 上の多くのユーザがそれぞれのコンテンツにタグと呼ばれる分類キーワードを付与することで，そのタグを基に分類やコンテンツへのアクセスを可能にする Folksonomy を構成している．このように，Web 上のユーザから広くアノテーションを収集することがアノテーションコストの低減に有効である．これらのシステムは Web 上でアノテーションを共有しており，そのアノテーションを元に様々な応用が実現されている．

コンテンツのセマンティクスに基づく応用のためには，コンテンツの論理構造の把握と，その構造への意味的属性の付与，つまりアノテーションが必要である．文章の場合はその文章の段落や構文情報などが，音楽の場合は，音符・歌詞などの楽譜情報や，楽曲パートの情報，イントロ・サビなどの楽曲の意味的な区間などがコンテンツの構造に相当する．これらの情報を扱うためには，コンテンツの内部を指し示す形式が必要となるが，アノテーション共有の重要性からも，その形式は汎用性が求められる．XML ドキュメントに対する内部へのポイント手法として XPointer[10] が，また連続メディアの時間的な区間を指し示す形式として URI time interval specification[11] が提案されているが，コンテンツの内部構造に対する一般的なポイント手法は存在しない．そのため音楽や画像などの構造化とその内部への言及を行う上での支障となっている．音楽のアノテーションシステムである MUCOSA[4] や CLAM Annotator[5] においては，楽曲内の時間的なセグメントに対してイントロやサビなどの楽曲構造情報や和音情報を付与を可能にしている．また後藤[3] は音響解析によって得られるビート構造やメロディーライン，バスラインにおける，時間区分に対するアノテーションを実現している．しかしこれらのシステムでは，いずれも楽曲の内部を指し示す形式が各システムに依存しており，アノテーションの表現形式自体が汎用性を持ち得ない．

どのような種類のアノテーションが必要であるかは，どのような応用を実現したいかに依存するため，事前に全ての種類のアノテーションを定義することができない．そのためアノテーションの形式は，柔軟にその定義を拡張できる必要がある．楽曲の基本情報を扱う形式として CDDB[12] が挙げられるが，これはアーティストや楽曲タイトルなど，定められたプロパティのみを扱う枠組みであり，例えばアプリケーション開発者が「その楽曲をどのような状況下で聴くべきか」という情報があれば新しい応用が実現できると考えても，新しいプロパティを付け加えることができない．また映像や音響信号に関するアノテーション形式として MPEG-7[13] が提案されている．しかし MPEG-7 によって詳細な音楽の情報を記述することは困難である．例えばあるプロモーションビデオに対して，MPEG-7 により楽譜に相当する音符や休符などの楽譜情報を記述するための語彙が用意されてはいない．一方 MPEG-7 では，あるセグメントに対しテキストを関連付けることができるため，それを利用して楽譜情報を入力することは可能であるが，それが楽譜情報であることを定義し，その定義を共有することができず，汎用性が低い．

また実際にアノテーションを収集するためには，アノテーションを記述するためのエディタが必要となる．従来のアノテーションエディタはアノテーションの定義と強い結びつきがあり，特定のアノテーションのみを収集するために構築されている[1]．しかしアノテーションの種類ごとに別々のエディタを構築するのは非効率であり，柔軟性が低い．

本研究では，これらの問題点を解決するため，任意のコンテンツに適用可能なアノテーションプラットフォーム Annphony[14] を構築し，また複数の解釈が存在するコンテンツの典型例である音楽に着目し，柔軟に音楽に対するアノテーションを取得する音楽アノテーションシステムを構築した．さらに収集されたアノテーションを複数のアプリケーションに適用した．本稿では，2 章で Annphony について，3 章で音楽アノテーションシステムについて述べる．4 章では音楽アノテーションシステムにより，多様な解釈を顕在化することが可能であることを確認する実験について述べ，5 章ではアノテーションに基づくアプリケーションである楽曲検索システムとプレイリスト作成支援システムについて述べる．

2. アノテーションプラットフォーム Annphony

楽曲に対する複数の解釈を管理するために，任意のデジタルコンテンツに対するアノテーションを扱うプラットフォームである Annphony[14] を構築した．以下に概要を述べる．

2.1 コンテンツの内部構造へのポイント手法

コンテンツの内部構造に対するアノテーションを扱うためには，そのコンテンツのセグメントを指し示す手段が必要である．特定のコンテンツの詳細箇所を指し示す形式として，XPointer[10] や URI time interval specification[11] などが存在するが，全てのメディアを網羅できておらず，また例えば「MIDI の特定のチャンネルにおける時間範囲」といった，提案されている形式では指し示すことのできない例が多く存

在する．そこで，XPointer の書式を参考にし，任意のメディアの任意のセグメントを指し示す形式として ElementPointer を提案する．

まず XPointer の例を示す．XPointer では，XML ドキュメント内部ノードを指し示す手段として標準化されている XPath[15] を用い，コンテンツの URI[16] に加え，#以降のフラグメント識別子として内部構造への言及を可能にしている．例えば XML ドキュメント (http://foo/bar.xml) の，あるノードを XPath(/document/node) により指し示す XPointer は以下のように記述される．

http://foo/bar.xml#xpointer(/document/node)

一方で，一般にはコンテンツの内部を指し示す XPath に相当する形式が存在しない．そこで ElementPointer ではコンテンツ内部を指し示すための定義を別途用意し，その定義を引用する．ElementPointer は以下の形式をとる．

[C]#epointer([S](([P1],arg1),([P2],arg2)...))

コンテンツの URI である [C] に続き，以降にフラグメント識別子としてコンテンツのどの部分を指し示すかを記述する．本手法では，コンテンツの内部は，[S] で表される ElementPointer 定義の URI 以降に，[P] で表される有限個のプロパティの URI とその値を列挙することにより指し示される．実際にはフラグメント識別子は URI としての妥当性を保つため，URL エンコーディングを行う．

アノテーション共有の必要性から，ElementPointer の定義は共有されることが求められるため，共有に適した一般的な形式で記述されるべきである．そこで，リソース間の関係を表すためのフレームワークであり，一般的なアノテーション形式として知られている RDF[17] のスキーマ言語である RDFS[18] を ElementPointer の定義に採用した．リソース間の関係をグラフ構造として表現する RDF や RDFS を用いることで，その検索言語である SPARQL[19] によって多彩なグラフ検索が可能になる．そのため ElementPointer の定義を共有することに適していると考えられる．Annphony においてその定義の検索や利用を支援し，任意のユーザによる ElementPointer の定義の利用を支援する．

ElementPointer の具体例を示す．前述の例である「MIDI の特定のチャンネルにおける時間範囲」を指し示すためには，MIDI のチャンネル番号，開始時間，終了時間という 3 つの値が決定される必要がある．そこで，MIDI チャンネル別の時間範囲を表す ElementPointer を定義する．本定義で利用されるプロパティとそのデータ型として，MIDI のチャンネル番号 (1 から 16 までの integer 型)，時間区分の開始時間 (double 型)，終了時間 (double 型) を記述する．また時間区分の開始時間，終了時間は楽曲の演奏の開始からの秒数であることを同時に記述する．具体的な RDFS の書式については 2.3 節において述べる．こうして記述された定義は Web 上に公開するか，Annphony に登録し，URI を持たせることで利用可能になる．便宜的に「MIDI の特定のチャンネルにおける時間範囲」に対する ElementPointer の定義の URI を [S]，チャンネル番号，時間区分の開始時間，終了時間の各プロパティの URI は [C]，[F]，[T] と表現すると，ElementPointer により，ある MIDI データ (http://foo/bar.mid) における，チャンネル番号が 1 番の，10.0 秒から 20.0 秒までの範囲を指し示す ElementPointer は以下のように表現される．

http://foo/bar.mid
#epointer([S](([C],1),([F],10.0),([T],20.0)))

また，例えば画像の矩形範囲を指定するためには，画像の X 座標，Y 座標，幅，高さが決定される必要がある．そこで同様に X 座標，Y 座標，幅，高さの 4 つのプロパティを記述し，さらにそれぞれの値のデータ型を指定する．またそれぞれの値は，画像の左上を頂点とし，それぞれの数値がピクセル単位であることを記述することで，画像の矩形範囲を表す ElementPointer を定義することができる．

Annphony は ElementPointer を扱う機能を備える．ElementPointer 定義の URI と対象コンテンツの URI を指定し，また ElementPointer の定義に沿ったプロパティの値を全て設定することで ElementPointer URI が生成される．逆に ElementPointer の URI から，コンテンツの URI や各引数の値を取得することも可能である．

2.2 構造と多様な解釈のためのアノテーション

リソースの関係を有向グラフで表現するアノテーション形式として RDF が提案されているが，RDF は一意の解釈や定義の記述を目的とした形式であり，多様な解釈を記述することに適していない．一方 RDF を拡張した形式として提案されている Named Graphs[20] は，任意のグラフのまとまりを一意に識別し，他と区別するために，グラフのまとまりに識別子を付与することができる．RDF では，「この Web ページは (主語) A さんに (述語) 作成された (目的語)」というように，主語・述語・目的語の Triple で記述されるが，Named Graphs はそれに加え，そのアノテーションの識別子であるアノテーション URI を持つ Quad で表現される．本形式は一つ一つの解釈を区別することができることから，複数の解釈を扱う形式として適していると考えられる．そこで Annphony では Named Graphs

をアノテーションの記述形式に採用した．Web 上で普及している URI の枠組みによりアノテーションを指し示すことができるため，アノテーションの再利用性が高まり，任意のアプリケーションによるアノテーションの利用を可能にする．

Annphony が扱うアノテーションは人により記述されるため，人が記述した不十分な解釈を補足するさらなる解釈が必要になるだろう．しかしどのようなアノテーションに対してさらなる言及が必要であるかを事前に予測することは不可能であるため，あらかじめ最小単位のアノテーションに対して識別子が付与される必要がある．そこで Annphony では任意のグラフのまとまりに対して識別子を付与することができるという Named Graphs の機能に制限を加える．Named Graphs では，図 1 の左側のように [A]，[B]，[C] というアノテーションをまとめたものに対して一つの識別子を付与することができるが，この場合識別子が付与されていない [A]，[B]，[C] の各部分への言及が困難である．Annphony では，図 1 の右側のように，それぞれのリソース記述を分離する．具体的には，あるアノテータが，ある主語 (リソース) に対して述語・目的語の組を列挙したものを一つのアノテーションとし，それぞれのアノテーションに URI を発行する．

音楽のプレイリスト中に含まれる複数の楽曲を対象としてコメントを付与する場合など，複数の楽曲という単一ではないリソースを主語としたアノテーションが想定される．そこで Annphony ではアノテーションの主語として複数のリソースを指定することを許可している．また図 1 の左側において [A]，[B]，[C] が何らかの意図を持ってまとめられていたということも，[A]，[B]，[C] という複数のリソースを主語とするアノテーションにより表現可能である．

コンテンツの代表的な構造として，グルーピング構造，ツリー構造，グラフ構造が挙げられる．RDF はリソース間の関係をグラフ構造で表現する形式であり，グラフ構造はツリー構造を内包するため，RDF はこれらの構造を扱うことのできる形式であるといえる．Annphony ではそれに加え，前述の ElementPointer によるコンテンツのセグメント指定と，主語に複数のリソースの記述を許すアノテーション形式により，コンテンツのグルーピング構造を表現することができる．例えば楽譜中に表れる音符や休符群をまとめて，そのグループに対してイントロやサビなどの楽曲構成情報などを関連付けることが可能になる．そのため Annphony はコンテンツの代表的な構造を扱うことができる形式であるといえる．

2.3 スキーマに基づくアノテーション管理

従来のコンテンツの内部構造に言及するアノテーション記述形式[21)13)] は，アノテーション記述のための全

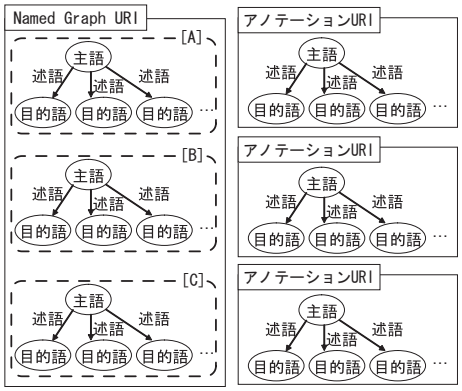


図 1 Named Graphs(左) と Annphony アノテーション (右)

```
<xsd:simpleType name="impressionType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="陽気"/>
  <xsd:enumeration value="刺激的"/>
  <xsd:enumeration value="厳格"/>
 </xsd:restriction>
</xsd:simpleType>
```

図 2 データ型の例

ての仕様が固定されており，拡張性が低い．そのためこれらの形式で記述することのできないアノテーションを利用することが困難である．そこで，Annphony のアノテーション形式のベースである RDF のスキーマ言語である RDFS を，アノテーション定義のための言語に採用した．取得すべきアノテーションの形式を RDFS により記述することにより，Annphony はそれに基づく検索やアノテーション利用を支援する．

アノテーション定義の記述方法を以下に述べる．例として，事前に決められた「陽気」「刺激的」「厳格」という 3 つの印象語のいずれかを値として持つ「印象情報」という名前のアノテーションを定義する場合を挙げる．

まずこれら 3 つの印象語のうちいずれかの値のみを許可するデータ型を定義する．データ型の定義は XML Schema[22)] により記述される．XML Schema は XML 文書の構造を定義するスキーマ言語の一つであり，柔軟にデータ型を拡張することができる特徴を持つ．一般的に RDFS では XML Schema において定義されるデータ型が利用されている．図 2 に実際のデータ型の定義を示す．XML Schema において定義された語彙を **xsd:** という接頭辞を用いて利用する．データ型の拡張には **simpleType** を用いる．この例では XML Schema で定義された基本データ型である string 型を **xsd:restriction** という語彙で拡張している．**xsd:enumeration** により列挙された 3 つの語の中から一つを選択するためのデータ型が定義されている．

```
<rdfs:Class rdf:about="http://foo/newschema.rdfs#impression">
 <rdfs:label>印象情報</rdfs:label>
 <rdfs:comment>定義の説明</rdfs:comment>
</rdfs:Class>
<rdf:Property rdf:about="http://foo/newschema.rdfs#impression_term">
 <rdfs:label>印象語</rdfs:label>
 <rdfs:comment>プロパティの説明</rdfs:comment>
 <rdfs:domain rdf:resource="http://foo/newschema.rdfs#impression"/>
 <rdfs:range rdf:resource="http://foo/datatype.xsd#impressionType"/>
</rdf:Property>
```

図 3 アノテーション定義の例

次に，これらの印象語を値として持つアノテーションの定義を図 3 のように記述する．RDF，RDFS により定義されている語彙は，**rdf:**や **rdfs:**という接頭辞を用いて利用している．RDFS において新しいアノテーション定義を行うために，まず **rdfs:Class** を記述する．属性として，その定義を利用するために必要な URI を，**rdf:about** という属性において記述する．**rdfs:label** にはその定義の名前を，**rdfs:comment** にはさらに詳細な説明を自然言語で記述する．次に，**rdf:Property** において，利用可能なプロパティを定義する．プロパティにも同様に，プロパティの URI，名前，説明を記述する．また，どの定義に対してこのプロパティが適用されるかを **rdfs:domain** の属性である **rdf:resource** において指定する．さらに，そのプロパティがどのようなデータ型であるかを **rdf:range** において指定する．ここで指定されているデータ型の URI は，図 2 において定義した 3 つの印象語からいずれかを選択するデータ型を指定している．以上で印象情報のアノテーションが定義された．

上記のアノテーション定義に基づいた実際のアノテーションの例を図 4 に示す．**an:**は Annphony で定義されている語彙を，**def:** は上記のアノテーション定義を利用するための接頭辞である．この例では先ほど定義した印象情報アノテーションを表す **def:impression** が指定されている．属性である **rdf:about** の URI は，Annphony が自動的に発行するアノテーションの識別子である．アノテーションの対象となるリソースは **an:target** に，また，アノテータの URI は **an:annotator** において指定されている．楽曲そのものに対してのアノテーションである場合は，**an:target** としてその楽曲の URI を，ある楽曲の区間に対するアノテーションの場合，2.1 節で述べた ElementPointer の URI を記述する．またそれらのリソースの集合を対象とする場合，図 5 のように，RDF のコレクション表現のための形式である **rdf:Seq**(順序付リスト)，**rdf:Bag**(順序無し集合)，**rdf:Alt**(代替表現リスト)を用いて複数の対象を列挙する．印象語を記述するプロパティである **def:impression_term** には，図 2 において定義された範囲の値を記述することができる．

このように Annphony はアノテーション定義に基づくアノテーションを管理する．アノテーション定義は任意のユーザにより記述することができるため，Annphony によりコンテンツの内部構造や解釈に対する幅広い種類のアノテーションを管理することができる．2.1 節で述べた ElementPointer の定義も，本アノテーション定義と同様の形式で記述される．

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:an="http://foo/annotation.rdfs#"
    xmlns:def="http://foo/newschema.rdfs#">
  <def:impression
   rdf:about="http://foo/annotation?key=1">
    <an:target rdf:resource="http://bar/tune.mp3"/>
    <an:annotator rdf:resource="http://bar/user.xml"/>
    <def:impression_term>厳格</def:impression_term>
  </def:impression>
</rdf:RDF>
```

図 4 Annphony におけるアノテーションの例

```
<an:target>
 <rdf:Seq>
  <rdf:li rdf:resource="http://bar/resource1.obj"/>
  <rdf:li rdf:resource="http://bar/resource2.obj"/>
 </rdf:Seq>
</an:target>
```

図 5 複数のアノテーション対象の指定

後述する音楽アノテーションシステムでは，複数の種類のアノテーションエディタを備えており，それぞれのアノテーション定義がどのエディタに適用可能であるかを判断する必要がある．そこで，各アノテーション定義がどのアノテーションエディタで利用可能であるかという情報を，アノテーション定義へのアノテーションとして記述し，Annphony に登録する．

2.4 Annphony の実装環境

実装環境を表 1 にまとめる．Annphony は様々なアプリケーションやアノテーションエディタによるアノテーションやアノテーション定義の登録や検索の要求などをブラウザなどの Web クライアントから受け付けるため，Web サーバとして実装されている．Annphony は XML-RPC(XML-Remote Procedure Call)[23] によって要求を取得し，検索結果の送信などを行う．また，クライアント側は Annphony API を用いることでサーバへの要求や，アノテーションの生成を行う．RDF を Java で扱うための API として，Java によるセマンティック Web アプリケーション開発のためのフレームワークである Jena[24] を利用している．

実装言語	Java (JDK 1.5) Java Servlet (Apache Tomcat 5.5)
HTTP サーバ	Apache HTTP Server 2.1
使用ライブラリ	Jena 2.2
データベース	PostgreSQL 8.1
対応環境	Windows, Linux

表 1 Annphony の実装環境

2.5 Annphony と各システム間の関係

図 6 に Annphony と後述する音楽アノテーションシステム，アプリケーションとの関連を示す．アプリケーション開発者は収集するべきアノテーションの定義を行い，Annphony に登録する．既に必要なアノテーションの定義が別アプリケーションの開発者により登録されている場合は新たに定義を行う必要は無い．次章で述べる音楽アノテーションシステムの各アノテーションエディタは，自身に適用可能であるというアノテーションの付与された全てのアノテーション定義を Annphony から受け取る．アノテーションエディタは取得した定義群から，それぞれの定義がどのようなデータ型のプロパティを持つかを Annphony API を利用して解析し，さらに，ユーザが入力可能なアノテーションメニューを動的に構成する．メニューの動的な構成法はアノテーションエディタにゆだねられている．例えばアノテーションエディタは，HTML のフォームを自動生成したり，右クリックメニューを変更するなどの処理を行う．ユーザからのアノテーションの入力を音楽アノテーションシステムのサーバが受け取ると，Annphony にアノテーションを登録する．アプリケーションは，アノテーションエディタを用いて収集されたアノテーションを Annphony を通して取得し利用する．

Annphony においてアノテーションとその定義を管理することで，従来強い依存関係にあったアノテーション定義とアノテーションエディタ，またアノテーションとアプリケーションをそれぞれ独立させる．Annphony において管理されるアノテーション定義を参照し，動的にアノテーションエディタを構成することで，柔軟なアノテーションの取得を見込むことができる．また Annphony において管理されるアノテーションはアプリケーションを問わず利用することができるため，類似するアノテーションを重複して収集する必要がなくなり，アノテーション作成のコストを抑えられる．

Annphony は音楽以外のコンテンツへのアノテーションも同時に管理することができる．音楽のアプリケーションのために収集されたアノテーションを，別のコンテンツのアプリケーションが利用するなど，コンテンツの種類を横断するアプリケーションの実現を支援するプラットフォームであるといえる．

3. 音楽アノテーションシステム

音楽に関する多様な解釈を取得するためのシステムとして，音楽アノテーションシステムを構築した．本システムは，Annphony において管理されるアノテーション定義に基づいてエディタを動的に構成するため，前章でも述べたように収集するアノテーションの種類を限定せず，幅広いアプリケーションで利用可能なアノテーション収集を支援する．

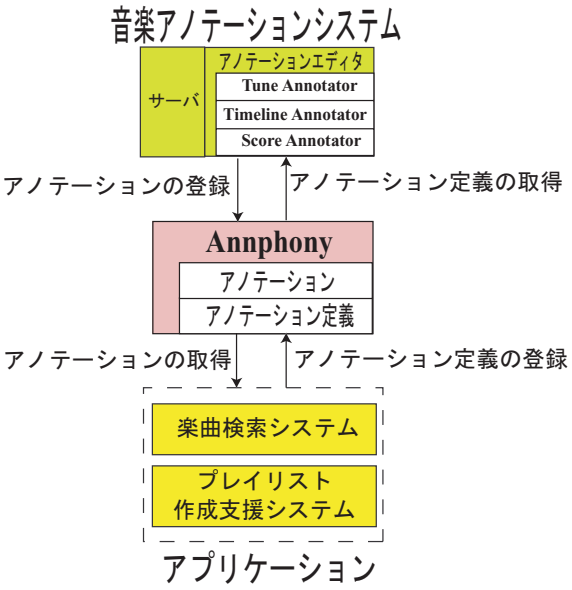


図 6 Annphony と各システム間の関係

本システムは Web サービスとして構築されており，Web 上のユーザからのアノテーション収集を実現している．多数のユーザから少量のライトウェイト・アノテーションを収集することにより，一人一人にかかる人的コストを低く抑える．

3.1 粒度の異なるアノテーション取得のためのエディタ

様々な種類のアノテーションを取得するために，対象となるリソースの粒度に応じたエディタを構築する必要がある．楽曲に対するアノテーション付与の代表例として，CDDB[12] や MusicBrainz[25] が挙げられる．これらのように，書誌情報などの基本情報を楽曲自体に対するアノテーションが必要であり，一方でコンテンツのセマンティクスに基づく応用のためには，コンテンツの内部構造に関するアノテーションが必要である．

そこで我々は，楽曲の形態や取得するアノテーションの種類に応じて様々な粒度のアノテーション取得を支援するため，1. Tune Annotator，2. Timeline Annotator，3. Score Annotator の 3 種類のエディタを構築した．以下にそれぞれの詳細を述べる．

Tune Annotator

Tune Annotator は書誌情報など楽曲自体に関する情報を収集するためのエディタである．タイトルやアーティスト情報などの，楽曲の基本情報の記述にとどまらず，楽曲の推薦度やアンケートなどの柔軟なアノテーション取得を支援する．Tune Annotator の画面には，

タイトルやアーティストなどの楽曲の基本情報，再生用のプレイヤと同時に，後述する Timeline Annotator と同様の形式のフォームが表示される．ユーザがアノテーションの種類を選択すると，Tune Annotator は自動的に該当するアノテーションのプロパティの入力フォームを構成する．ユーザは各項目に入力することでアノテーションを行う．

本エディタを利用するための条件は，楽曲が URI で表現されることである．つまり，Web 上に存在する全てのメディア形式の楽曲に対して Tune Annotator を利用したアノテーション収集が可能である．

Timeline Annotator

楽曲の最も単純なセグメンテーション手法は時間による区分であり，エンドユーザにとって直感的に理解することができる．そこで，MP3 などの連続メディアに対する時間的なセグメンテーションと，セグメントに対しアノテーションを付与するエディタである Timeline Annotator を構築した．図 7 に Timeline Annotator の画面例を示す．上部の音楽プレイヤで楽曲を鑑賞しながら，[FROM][TO] ボタンにより開始，終了時間を指定することで時間的なセグメントを切り出す．さらに [ANNOTATE] ボタンを押すと，図 8 に示すアノテーションフォームが現れる．ユーザはアノテーションの種類を選択すると，アノテーションのプロパティの入力フォームを自動的に構成する．ユーザが各項目に入力することでアノテーションを行う．図 8 に例示した音符情報は，3.2 節において連続メディアの構造化を行うためのアノテーションの一つである．

付与されたアノテーションは，図 7 の下部にリンクとして表示され，リンクにマウスポインタを当てることで内容の確認を行うことができる．

楽曲の時間区分の表現形式として，前章で述べた ElementPointer を利用している．セグメントの開始時間，終了時間という二つのプロパティを持つ ElementPointer を定義することで，楽曲の時間的な区間を URI により表現することが可能になる．Timeline Annotator はこの連続メディアの時間区分を表す ElementPointer URI を対象としたアノテーションを記述することで，楽曲の内部へのアノテーションを実現している．

本エディタを利用するためには，1. 楽曲が URI を持ち，2. 音楽プレイヤで再生可能な連続メディアである必要がある．広く普及している MP3 や MIDI などのメディア形式に対応するため，実用性の高いエディタであるといえる．

Score Annotator

Score Annotator は，楽譜に現れる要素の集合に対するアノテーションを付与するためのエディタである．アノテーションの付与が可能な楽譜の要素は，各楽器

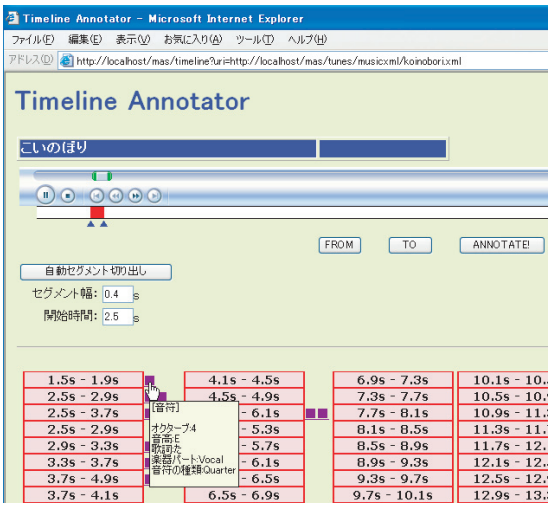


図 7 Timeline Annotator の画面例

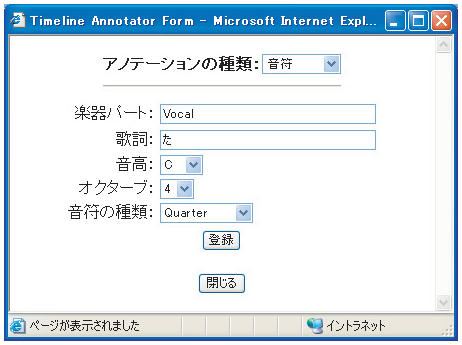


図 8 Timeline Annotator のアノテーションメニュー例

パートの音符，歌詞，タイトル，作詞者，作曲者，発想記号である．ユーザには図 9 のように楽譜とアノテーションを可視化したオブジェクトが重ねて表示される．ユーザがブラウザに表示された楽譜から，マウスのドラッグにより矩形の範囲指定を行うと，矩形内に存在する要素または要素集合が選択され，赤色で強調表示される．複数の矩形範囲を同時に選択することも可能であるため，楽譜上で遠い位置に存在する要素に対して同時にアノテーションを付与することができる．次に図 10 のように，矩形上でマウスを右クリックし，出現するメニューからアノテーションの種類を決定し，その後具体的な情報を記述する．

感想や解説などのアノテーションは，人間が記述する情報であるために，それが不十分であったり誤っている場合があるだろう．2.2 節で述べたように，Annphony で管理されるアノテーションには，全て識別子が自動的に付与されるため，アノテーションに対するアノテーションを表現することができる．しかしアノテーションエディタにもアノテーションへのアノテーションを行う仕組みが必要である．そのため本エディ

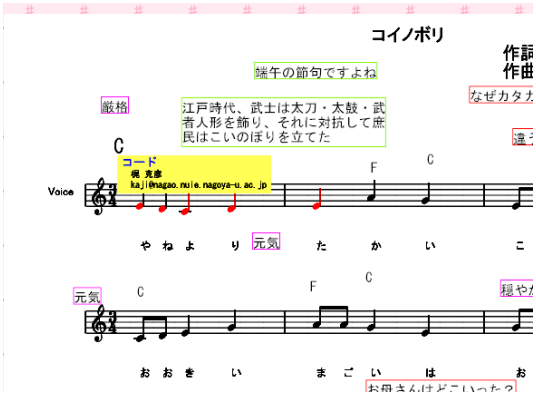


図 9 Score Annotator の画面例

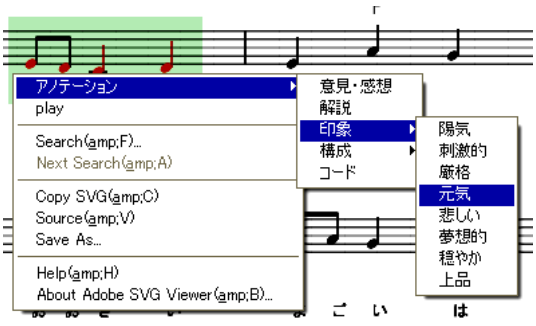


図 10 選択したオブジェクトへのアノテーション付与

タは，アノテーションを補足，修正するために，画面に出現するアノテーションオブジェクトに対してアノテーションを関連付ける機能を持つ．

本エディタを利用するためには，1. URI を持ち，2. メディア形式が MusicXML[26] である必要がある．MusicXML は楽譜を形成するために十分な情報が記述されており，楽曲の構造を表現する形式であるといえる．音楽的に構造化されたメディアに対するアノテーションを行うことにより，楽曲そのものや時間区分といった大まかな対象ではなく，楽譜という音楽の論理構造に基づいた詳細な対象への言及を実現している．音楽的に意味のある箇所へのアノテーションにより，楽曲のセマンティクスに基づく，より高度な音楽アプリケーションの実現を期待できる．

3.2 連続メディアの構造化支援

前節で述べたように，音楽アノテーションシステムは楽曲のセマンティクスに基づく応用のため，Score Annotator によるアノテーション付与を実現しているが，本エディタは MusicXML という楽曲の論理構造を表す形式でなければ利用できないという大きな制約がある．現在 Web 上に存在するメディアの多くは MP3 などの連続メディアであり，音楽のセマンティクスを考慮したアプリケーション実現のために，それらのメディアを構造化する必要がある．そこで連続メディアに対する構造化の支援を行う機能を構築した．

楽曲の構造化を行うために，タイトルやアーティストなどの基本情報に関するアノテーション定義に加え，楽譜生成のために十分な情報をアノテーションとして記述するための，楽器パートや発音列 (音高，音長，音符のタイプ)，歌詞などのアノテーション定義を用意した．また，それぞれのアノテーション定義を Tune Annotator と Timeline Annotator に適用した．ユーザは楽曲の基本情報を Tune Annotator を用いて記述する．次に Timeline Annotator を用いて，構造化を行う連続メディアを時間的にセグメントし，各セグメントの音符情報や小節の情報を入力する．一つの音符や歌詞の一単語など，一つ一つのアノテーションは楽曲の断片情報を表現しているに過ぎないが，ある楽曲に付与されたこれらのアノテーションの集合から，楽曲全体の楽譜を生成することができる．

次に，音楽アノテーションシステムはこれらのアノテーション集合を解析し，MusicXML とそれに対するアノテーションを以下のように生成する．アノテーション集合から読み取ることができる，メロディや小節の情報を元に，その楽曲を表す MusicXML を生成し，同時に MusicXML に現れる音符や小節に関する各ノードと，それに対応する連続メディアの時間区分を対応付けるアノテーションを生成する．

連続メディアを構造化し，MusicXML を生成することにより，Score Annotator への適用が可能になる．本機能により，連続メディアに対する，従来困難であった論理構造に基づく応用を実現することができるだろう．

連続メディアの構造化には音楽的な知識が必要なため，専門性が高い．そのため，連続メディアの構造に関するアノテーションは Web 上から幅広く収集することが困難であると予想される．本システムでは人間が作成するアノテーションに加えて機械が自動的に生成する情報も扱うことができるため，ビートトラッキング[27] などの採譜に関する自動処理の結果を随時取り込み，その結果を人手により必要に応じて修正することでアノテーションの作成コストを下げることが必要になる．

4. 多様な解釈の顕在化に関する実験

ユーザや楽曲の特性によって解釈に多様性が存在し，個人に適応するアプリケーションを実現するために解釈の多様性と個人性を扱う必要があること，またそのような解釈の多様性を音楽アノテーションシステムによって顕在化させることが可能であることを実証するため，「ハイライト」の認識に関する実験を行った．楽曲のどの部分を，どのような理由でハイライトと認識

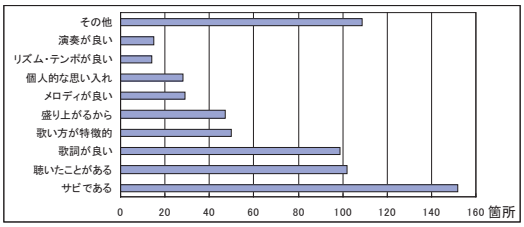


図 11 理由ごとのハイライト区間数

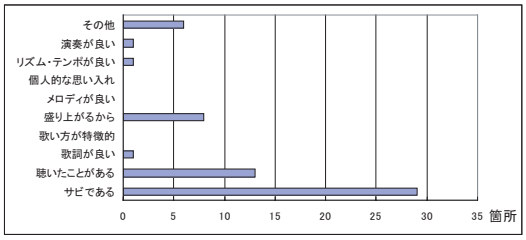


図 12 ユーザ A における理由ごとのハイライト区間数

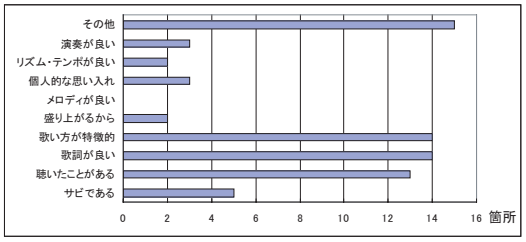


図 13 ユーザ B における理由ごとのハイライト区間数

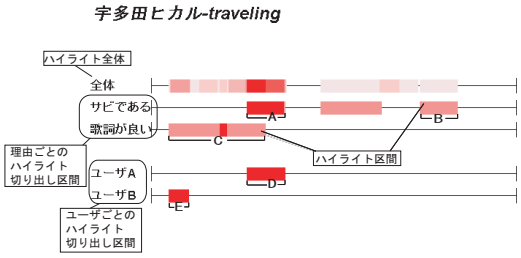


図 14 ハイライトと認識された区間の分布

するかが解釈に相当する．本実験では，ハイライトの定義を「楽曲の最も印象的な部分」とした．以下に実験の詳細を述べる．またハイライトと認識する理由の一つに「サビである」という理由が予想されるが，本実験ではサビの定義を「楽曲の中で繰り返しの最も多い区間」としている．

実験に利用する楽曲は，過去 5 年におけるオリコン年間売り上げの上位 10 曲ずつ，計 50 曲であり，被験者は 20 代の男女 10 名である．それぞれの被験者は，時間区分に対するアノテーションエディタである Timeline Annotator を用い，各楽曲について，ハイライトであると認識した時間区分を切り出し，同時にその部分をハイライトであると認識した理由を自然言語により記述する．ハイライト区間は，各楽曲につき一箇所であり，切り出しの長さに制限を設けていない．50 曲の楽曲に対して 500 箇所のハイライト区間と，645 個の理由を収集した．ハイライトであると認識した理由には複数の記述を許しているため，理由の合計は 500 個を超えている．

収集されたアノテーションを基に，ハイライトと認識した理由を人手で分類したところ，以下の 9 種類に分類された．

- サビである
- 聴いたことがある
- 歌詞が良い
- 歌い方が特徴的
- 盛り上がるから
- メロディが良い
- リズム・テンポが良い
- 個人的な思い入れ
- 演奏が特徴的

図 11 はハイライトと認識した理由ごとのハイライト区間の数を表している．従来からハイライトであることが多いとされる「サビである」という理由以外にも,「聴いたことがある」「歌詞が特徴的」「歌い方が特徴的」という理由に基づく認識が多いことが分かる．

次にユーザごとのハイライト認識理由の個人性について考察する．図 12 にユーザ A，13 にユーザ B がハイライトであると認識した理由の割合を示す．ユーザ A は「サビである」「聴いたことがある」という理由でハイライトであると認識された箇所が多く，一方ユーザ B は「歌詞が良い」,「歌い方が特徴的」という理由でハイライトであると認識する可能性が高いことが読み取れる．このようにユーザごとのアノテーションを解析することにより，それぞれのユーザの個人性を浮き上がらせることができることを確認した．

次に切り出されたハイライト区間についての考察を行う．図 14 に，ある楽曲のサビ区間と理由別のハイライトを可視化した図を示す．楽曲中で，多くの被験者がハイライトだと認識した区間ほど濃い色で表されている．一段目には理由に関わらずハイライトだと認識された区間を表示し，二段目にはサビであるという理由で，三段目には歌詞が特徴的であるという理由で認識されたハイライト区間を表示している．一段目から，ハイライトであると認識された区間は楽曲の多くの区間に渡っていることが分かる．また二段目と三段目には重複する区間が少なく，大きな違いがあることが分かる．このことからハイライトと認識する理由によって切り出される区間の多様性を，アノテーションの分類という統計的な処理によって浮き上がらせることができたといえる．

楽曲の解釈に関する多様性や個人性を利用することにより，それぞれのユーザに適応するアプリケーションの実現が可能になると考えられる．例えば楽曲のハイライトを試聴支援に用いる場合，サビを高い確率でハイライトと認識するユーザ A にとっては，図 14 の区間 A，B を中心に試聴させることが効果的であり，歌詞に基づきハイライトを認識するユーザ B にとっては区間 C を中心に試聴を促すことが効果的であると考えられる．実際に図 14 にはユーザ A が本実験で切り出したハイライト区間 D は，ユーザ A にとって試聴支援に効果的であると思われる区間 A，B に含まれており，またユーザ B のハイライト区間 E も同様に，試聴支援に効果的であると思われる区間 C に含まれている．

本実験により，楽曲の内部構造に関する解釈の多様性と各ユーザの個人性を，本システムで収集されるアノテーションを解析することによって浮き上がらせることが可能であること，またこうして導き出された個人性をアプリケーションに適用可能であることが確認された．

5. 楽曲の多様な解釈を用いたアプリケーション

音楽アノテーションシステムにより，様々なアプリケーションに適用可能なアノテーションを Web 上から収集することが容易になった．楽曲に対する多様な解釈を用いることで，ユーザの個人性に基づくアプリケーションを実現することができることを確認するために，楽曲検索システムとプレイリスト作成支援システムを構築した．

5.1 楽曲検索システム

楽曲の構造や多様な解釈のアノテーションを利用することで，個人の感性に合わせた楽曲の内部検索が可能になる．そこで我々は，サビやイントロなどの楽曲の論理構造に基づき，ユーザの感性に合わせて楽曲の印象検索を行うシステムを構築した．以下に本検索システムのために収集したアノテーションと，それぞれの検索機能について述べる．

検索システムのためのアノテーション定義

楽曲の論理構造に基づく検索のために，Score Annotator を用いて楽譜に出現する要素集合に対するアノテーションを収集した．アノテーションの定義として印象，楽曲の構成の 2 種類を設定した．

ユーザによる解釈のばらつきが多くみられ，個人に合わせた楽曲検索に利用可能であると考えられる情報として，楽曲を鑑賞した際に受ける印象が挙げられる．そこで，楽曲の各部分に対し，ユーザが楽曲を鑑賞して受けた印象を Hevner[28] が提唱した 8 種類の印象語 (陽気, 刺激的, 厳格, 元気, 悲しい, 夢想的, 穏やか, 上品) から選択し，収集することにした．

また，音楽の構造に基づく検索に有効であると考えられる，楽曲の構成の情報を収集する．楽曲の構成は，ポップスなどに見られるイントロやサビといった，その曲の大まかな構成を記述するためのアノテーションであり，試聴支援[29] などに利用される価値の高い情報である．

個人の感性に適応する検索

多様な解釈を利用し，個人の感性に合わせた検索機能として，印象に基づく楽曲検索機能を構築した．印象情報はユーザ間の競合がしばしば見られる情報であるが，類似する感性を持つユーザ同士は，同一楽曲やセグメントから類似する印象を受ける．そこで検索するユーザ自身がアノテーションを付与していない楽曲についても印象検索の対象にするために，類似するユーザが付与したアノテーションを利用する．本検索システムでは，検索するユーザと，ある閾値以上感性の類似するユーザが付与した印象アノテーションのみを利用し，楽曲検索を行う．それぞれのユーザがどのような感性であるかというユーザモデルとユーザ間の類似度の閾値は，後述するプレイリスト作成支援システムのものを用いた．

一般の楽曲検索の場合，検索結果のランクは検索要求との適合度のみによってランキングされるが，本検索システムではアノテータに合わせたアノテーションのフィルタリングを行うことで，例えばある二つの楽曲に，等しい数の「悲しい」という印象が付与されていた場合，嗜好の類似したユーザによるアノテーションが多い楽曲の方が上位にランクされる．ユーザの感性に合わせた検索結果のランキングを行うことで，それぞれのユーザが要求する楽曲を適切に提示する．

楽曲の構成に基づく絞込み検索

サビやイントロなどの楽曲の構成は，エンドユーザが容易に認識できる楽曲の内部構造であるといえる．そこで本検索システムに，楽曲の構成による絞込み検索の機能を実装した．図 15 に本システムの検索フォームを示す．ユーザは本検索システムを利用する際にログインが必要であり，システムは誰が検索をしているかを知ることができる．検索フォームにおいて，左側のリストボックスからは楽曲の構造を，右側のリストボックスから印象を選択することにより，例えば「サビが穏やかな曲」といった検索が可能になる．検索結果は図 16 のように，検索ランクが上位の楽曲から順に列挙される．

「サビが悲しい曲」という検索要求を受け取った場合,「サビ」が絞込みを行う楽曲の構成であり，検索対象は「悲しい」という印象である．まず，楽曲の構成以外の検索要求である「悲しい」について，前述の印象に基づく検索を行う．この時の検索結果のランクを

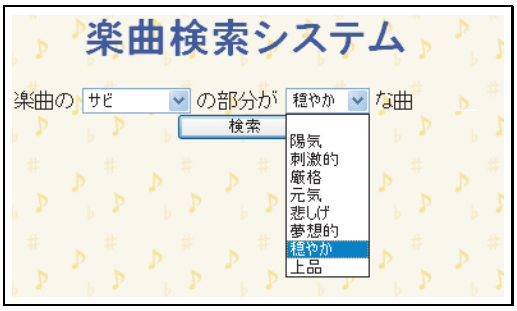


図 15 楽曲検索システムの検索フォーム

図 16 楽曲検索システムでの検索結果

計算する．この時の検索結果のランクを priorrank とする．次に，絞込む楽曲の構成「サビ」の部分に対する「悲しい」などの含有率 contains を計算し，含有率に基づき絞込み検索のランクを決定する．

「サビ」のアノテーションが付与されている音符，休符などの音楽オブジェクトが o_j から o_l の要素であり，その要素数は $l-j+1$ であるとする．また，「悲しい」というアノテーションが付与されている音楽オブジェクトは，o_i から o_k までで，その要素数は $k-i+1$ であるとし，これら二つのアノテーションが共通して付与されているオブジェクトは，o_j から o_k までで，その要素数は $k-j+1$ であるとすると，含有率 contains は以下の式で求める．

$$\text{contains} = \max\left(\frac{k-j+1}{k-i+1}, \frac{k-j+1}{l-j+1}\right)$$

「サビ」の部分に多く「悲しい」の要素が含まれている場合，または「悲しい」と関連付けられた要素が多く「サビ」に含まれている場合に contains が高くなる．最終的な絞り込みの検索ランクは以下の式で表される．

$$\text{rank} = \text{contains} * \text{priorrank}$$

本機能により，楽曲のセグメントの検索が可能となり，引用・要約など様々なアプリケーションにおいてそのセグメントを利用することができるようになる．検索結果として得られるセグメントは楽曲の構造に対するアノテーションに基づいているため，音楽的に意味のあるセグメントであるといえる．

本節で述べた個人適応された楽曲の内部検索は，楽曲要約や編曲など様々な音楽アプリケーションにおいて，利用する楽曲やその部分を選択する導入部として利用可能であると考えられる．

5.2 プレイリスト作成支援システム

ジャンルやアーティスト，歌詞など，一般的に，楽曲推薦に効果的であるとされている楽曲の特徴量がいくつか存在する[31)32)]．一方，日常生活における音楽との接し方から推測されるように，リスナは置かれている状況によって鑑賞する楽曲が変化する．海に遊びに行けば，楽しく明るい曲を選んで場を盛り上げ，恋人と二人でいる時は，ゆったりとした幸せな曲を選んでムードを高める．そのため楽曲推薦システムは，リスナの嗜好に加え，置かれている状況に合わせた楽曲を推薦することが望ましい．そこで我々はリスナの嗜好と状況に合ったプレイリスト作成支援システム[30)]を構築した．

アノテーションの定義と収集

本システムは，楽曲の特徴量として歌詞・表現している情景 (楽曲情景)・鑑賞したい状況 (鑑賞状況) の 3 種類を採用している．楽曲情景は,「片思いの心象を歌った曲」や「卒業・別れの曲」など，その楽曲がどのような情景を歌っているかという情報である．鑑賞状況は,「海に大勢で遊びに来ている時」,「恋人と二人のクリスマスの夜」など，その楽曲をどのような状況で聴きたいかという情報である．楽曲情景，鑑賞状況に関する情報はリスナによる楽曲の解釈に強く関わる情報であるため，楽曲の音響情報を自動解析して得ることが困難である．そこで，音楽アノテーションシステムの Tune Annotator により各楽曲に対して楽曲情景と鑑賞状況の情報を収集した．

楽曲情景・鑑賞状況アノテーションの定義にあたり，事前の予備実験において，どのような楽曲情景，鑑賞状況が存在するかを調査し，いつ・どこで・どのような心理状態であるかという項目を設定した．例えば「いつ」に関しては，朝，昼，夜，春，夏，秋，冬などの項目が設定されている．Tune Annotator を用いてアノテーションを収集した．

楽曲・ユーザの特徴量

歌詞の TF*IDF[33)]，楽曲情景，鑑賞状況の 3 種類をそれぞれの楽曲の特徴量とした．l_{t_i}， c_{t_i}，s_{t_i} を歌詞・楽曲情景・鑑賞状況の各特徴量空間における楽曲 t_i の特徴ベクトルとする．l_{t_i}， c_{t_i}，s_{t_i} は以下のように列ベクトルで表される．

$$l_{t_i} = (f(t_i, x_1), f(t_i, x_2), ..., f(t_i, x_n))$$
$$c_{t_i} = (f(t_i, y_1), f(t_i, y_1), ..., f(t_i, y_o))$$

$$s_{t_i} = (f(t_i, z_1), f(t_i, z_1), ..., f(t_i, z_p))$$

TF*IDF によって求められたキーワードの数は n であり，x_j は j 番目のキーワードを表している．$f(t_i, x_j)$ は楽曲 t_i の歌詞中に，キーワード x_j が含まれていれば 1，含まれていなければ 0 となる．o，p はそれぞれ楽曲情景，鑑賞状況アノテーションに設定された項目数であり，y_j，z_j は，それぞれ楽曲情景，鑑賞状況アノテーションの j 番目の項目を表している．また $f(t_i, y_j)$，$f(t_i, z_j)$ は，楽曲 t_i に対する楽曲情景，鑑賞状況アノテーションとして，y_j，z_j の項目が選択されている場合は 1，選択されていない場合は 0 となる．

各特徴量空間における楽曲間の類似度は，特徴ベクトル同士のコサイン距離により得られる．またそ歌詞・楽曲情景・鑑賞状況の特徴量空間で得られた各コサイン距離の重み付き和を，最終的な楽曲間の類似度とした．

一方，各ユーザとプレイリストも各特徴量空間にマップされる．嗜好に適合するとシステムに申告された楽曲集合の特徴量の平均をそのユーザの嗜好に最も適合する楽曲であるとし，ユーザは楽曲情景・歌詞の特徴量空間にマップされる．さらにユーザ自身が現在置かれている状況をシステムに入力した時点で鑑賞状況の特徴量空間にもユーザがマップされる．プレイリストも同様に，プレイリスト要素である楽曲の鑑賞状況・歌詞ベクトルの平均値と，そのプレイリストが制作された状況から，それぞれの特徴量空間にマップされる．楽曲間の類似度計算と同様に，コサイン距離の重み付き和により，楽曲とユーザ間，ユーザとプレイリスト間などの類似度を算出される．

プレイリスト作成支援

プレイリスト作成支援の手順を説明する．ユーザは本システムに,「冬の夜，一人で考え事をしている」といった現在ユーザが置かれている状況を入力する．ここでユーザが入力する項目は，鑑賞状況アノテーションと同様の項目である．システムはまず過去に作成されたプレイリストの中から本システムの操作履歴からモデル化されるユーザの嗜好と，現在の状況を踏まえて，類似する嗜好・状況において作成された基プレイリストを選出する協調フィルタリングを行う．次に，選び出された基プレイリストに対し，よりリスナに適合させる調整処理を施す．本システムでは，基プレイリストの調整処理として，以前にユーザ自身が「嗜好に合わない」と申告した楽曲を取り除き，またプレイリストに含まれる楽曲中の 3 割を，ユーザが以前に鑑賞したことのない楽曲に入れ替える操作を行っている．ユーザには図 17 のようにプレイリストが提示される．リスナは提示されたプレイリストを鑑賞し，それぞれの楽曲に対して嗜好に合っているか，現在の状況に合っているかを，楽曲リストの左側に現れる「○」「×」を選択して申告する．また現在のプレイリストから，入れ替えを行いたい楽曲の「入替」のボックスをチェックする．システムはこのインタラクション情報を基に，チェックされた楽曲を別の楽曲に差し替え，さらにリスナの嗜好に適応したプレイリストに変換していく．

ユーザの嗜好モデルは，本システムとのインタラクション情報を用いて順次更新される．TF*IDF で得られるキーワード数が x である場合，歌詞の特徴量空間における初期状態の基底は，x 次元単位行列と同等の標準基底である．ユーザが嗜好に合っているとフィードバックした楽曲群と，合っていないとフィードバックした楽曲群を用い，各ベクトル空間の基底をユーザの嗜好に適応させる．そのためベクトル空間の基底はそれぞれのユーザについて異なる．以下は歌詞の特徴量空間における基底をフィードバックに応じて変化させるための式である．

$$\mathrm{b}_u \leftarrow \mathrm{normalize}(\mathrm{b}_u + \mathrm{diag}(\delta \frac{l_f - l_u}{|l_f - l_u|} - \delta \frac{l_d - l_u}{|l_d - l_u|}))$$

b_u はユーザ u の歌詞の特徴量空間における基底である．l_f はフィードバックによって嗜好に合っているとされた楽曲の歌詞特徴ベクトルの平均であり，l_d は嗜好に合わないとされた楽曲の歌詞特徴ベクトルの平均である．l_u は，ユーザ u がこれまでに嗜好に合っているとフィードバックした楽曲全体の歌詞特徴ベクトルの平均である．一回のフィードバックにおいてどれだけ基底を変化させるかを決定する変換レートは δ で表されている．本システムでは経験的に δ の値を設定している．diag は列ベクトルを対角行列に変換する関数である．

このように変化させた基底を用いて各特徴量空間の基底変換を行うことで，特徴量空間を嗜好に合った楽曲の方向に短縮し，嗜好に合わない楽曲の方向に拡張する．楽曲情景と鑑賞状況の特徴量空間についても同様に，アノテーションの項目数を y，z とした場合，y，y 次元単位行列を初期状態とし，ユーザからのフィードバックごとに基底をユーザに適応させる．ユーザ間の類似度計算の際の基底変換には，プレイリストを作成する，または前述の楽曲検索システムで検索をする主体となるユーザの基底を用いる．

本システムは文献[30] における評価実験により，ユーザのフィードバックが得られれば得られるほど，適切なプレイリストを推薦することが確認されている．これらは音楽アノテーションシステムにより収集されたアノテーションを用いて，個人に適合する音楽アプリケーションを実現した例といえる．

入替	好きな曲？	状況に合ってる？	Playlist
☐	○	○	奥田民生 – イージューライダー'97
☐	×	○	aiko – 天の川
☐	○	○	Mr.Children – 星になれたら
☑	×	×	松任谷由実 – やさしさに包まれたなら
☐	○	○	aiko – ボーイフレンド
☑	○	×	Mr.Children – つよがり
☐	○	○	aiko – 白い服黒い服
☐	○	○	aiko – 桃色
☐	○	○	MIT – HORO
☐	○	○	Jeff Manning – I think of you

図 17 ユーザに提示されるプレイリスト

6. 関連研究

6.1 MoodLogic

楽曲に対する多様な解釈を収集し利用する例として，MoodLogic[34] が挙げられる．MoodLogic は Web 上のユーザコミュニティと専門家によって楽曲に対するタイトル・アーティストなどの基本情報に加えて，テンポや印象情報を収集し，「アップテンポなロック」や「ロマンチックなブルース」など，印象情報に基づく検索やプレイリスト生成を支援する．

本研究と MoodLogic との比較について述べる．まず，MoodLogic ではユーザから収集した印象・ジャンルなどの情報を統計し，その楽曲の印象・ジャンルを一意に決定している．一方本研究では，管理される複数の解釈をどのように扱うかはアプリケーションにゆだねられている．MoodLogic と同様に，統計をとり一意の値を取得することもできるが，それぞれのアノテーションにはアノテータ情報が付与されているため，本研究のプレイリスト作成支援システムのように，楽曲の嗜好に関するユーザモデルを用いることで，あるユーザと類似するユーザが付与したアノテーションを優先的に利用するなど，ユーザごとに個人化したアノテーション利用が可能になる．

また MoodLogic では楽曲検索とプレイリスト生成支援にのみ利用されるアノテーションを収集しており，また他のアプリケーションがその情報を利用することは困難である．このことから，アノテーション定義の柔軟性と，アノテーション共有に関して問題があるといえる．一方本アノテーションシステムはアノテーション定義をアプリケーション開発者が定義することで，アノテーション収集のためのエディタが動的に構成されるため，様々なアプリケーションに適用するアノテーションを収集することができる．

6.2 CUIDADO

CUIDADO (Content-based Unified Interfaces and Descriptors for Audio/music Databases available Online)[35] は，オンライン音楽データベースの統一的なインタフェースを提案し，楽曲ブラウジング・検索・オーサリングの支援を目指すプロジェクトである．メタデータの記述形式は MPEG-7[13] を参考にしており，タイトルやアーティストなどの基本情報を収集し統計的なインデキシングを行い，さらにオーディオデータからテンポや音響パワーなどの特徴量を自動抽出し管理する．

アプリケーションの一つとして，音楽制作者のための検索・編集・処理ツールである Sound Palette が提供されている．ユーザはメロディやリズムに基づく類似楽曲検索で得られる楽曲ファイルをインポートし，自動でセグメント化された楽曲の断片を利用することができる．またテンポの異なる二つの楽曲をミックスさせる際に，自動的にテンポを同期させるなど，音楽のセマンティクスに基づいた編集を支援する．

このプロジェクトでは，楽曲の内部構造に関する情報は自動的に抽出されているが，楽曲編集に有効な情報は，必ずしも自動的に抽出できるとは限らない．そのため音楽アノテーションシステムのように楽曲構造に関するアノテーションを人手により付与する必要があると考えられる．また，本論文のシステムのように本質的に自動的に抽出できない楽曲の特徴を収集するためにはユーザによるアノテーションは不可欠だと思われる．

7. おわりに

本稿では，コンテンツに対する多様な解釈を扱う必要性を指摘し，任意のデジタルコンテンツに対するアノテーションとその定義の管理を行うプラットフォームとして Annphony を構築した．RDF の拡張と ElementPointer の導入により，コンテンツの内部構造に対する多様な解釈を記述することができる．

また音楽を題材として，Web 上のユーザから様々な解釈を獲得する音楽アノテーションシステムを構築した．本システムでは，多様な種類の音楽メディアに対応するために連続メディアの構造化をサポートし，楽曲のメディア形式と取得するアノテーションの種類に関連して，1:書誌情報などの楽曲自体へのアノテーションエディタ，2: 連続メディアに対するアノテーションエディタ，3: 楽曲の内部構造に対するアノテーションエディタの 3 種類のエディタを備え，様々な粒度のアノテーション収集を支援する．またアノテーション定義とアノテーションエディタを独立させることにより，様々なアプリケーションに適用可能なアノテーションを柔軟に収集する．

また本システムでは，現在までに 47 人のユーザにより 21 種類，1126 個のアノテーションを収集し，それらのアノテーションを利用したアプリケーションとし

て楽曲検索システム，プレイリスト作成支援システムを構築した．それぞれのアプリケーションでは，ユーザ間の類似度に基づくアノテーションのフィルタリングを行うことで，ユーザやアプリケーションの特性に合わせたアノテーションを利用している．

今後の課題としては，楽曲に対する多様な解釈の傾向を分析し，他者の解釈への依存度や解釈の変化の時間的推移などの特徴を明らかにし，多様な解釈を総括する解釈を導くこと，また音楽以外のコンテンツに対しても複数の解釈を収集し，コンテンツの種類を横断するアプリケーションを構築することなどが挙げられる．

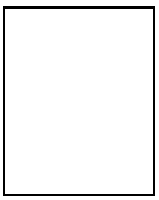
謝辞 本研究を進めるにあたり，貴重なご助言をくださいましたNTTコミュニケーション科学基礎研究所の平田圭二氏，また本論文を校正する上で数多くの重要なコメントをいただいた査読者の方々，そして，実験に協力していただいた名古屋大学長尾研究室のみなさまに深く感謝いたします．

参 考 文 献

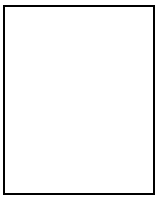
1) Nagao, K. : Digital Content Annotation and Transcoding, Artech House Publishers, (2003).
2) Smith, J. R. and Lugeon, B. : A Visual Annotation Tool for Multimedia Content Description, Proc. of the SPIE Photonics East, Internet Multimedia Management Systems, pp.49-59, (2000).
3) Goto, M. : Music Scene Description Project: Toward Audio-based Real-time Music Understanding, Proc. of ISMIR, (2003).
4) Herrera, P., et al. : MUCOSA: A Music Content Semantic Annotator, Proc. of ISMIR, (2005).
5) Amatriain, X., et al. : The CLAM Annotator: A Cross-platform Audio Descriptors Editing Tool, Proc. of ISMIR, (2005).
6) Dagan, I. and Engelson, S. P. : Committee-Based Sampling For Training Probabilistic Classifiers, International Conference on Machine Learning, pp. 150-157, (1995).
7) Mathes, A: Folksonomies - Cooperative Classification and Communication Through Shared Metadata, http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html, (2004).
8) del.icio.us. : http://del.icio.us/.
9) Flickr: http://www.flickr.com/.
10) W3C: XML Pointer Language, http://www.w3.org/TR/WD-xptr, (2001).
11) Pfeiffer, S., Parker, C. and Pang, A. : Specifying time intervals in URI queries and fragments of time-based Web resources, http://www.annodex.net/TR/URI_fragments.txt, (2003).
12) Gracenote: CDDB, http://www.gracenote.com/gn_products/cddb.html, (2003).
13) MPEG-7 Consortium: MPEG-7, http://www.mp7c.org/, (2002).
14) 梶 克彦, 長尾 確: 任意のデジタルコンテンツに対するアノテーションプラットフォーム, 電子情報通信学会 Web インテリジェンスとインタラクション, WI2-2006-16, pp.89-94, (2006).
15) W3C :XML Path Language, http://www.w3.org/TR/xpath.html, (1999).
16) Berners-Lee, T., Fielding, R. and Masinter, L. : Uniform Resource Identifiers (URI): Generic Syntax, RFC2396, (1998).
17) W3C: Resource Description Framework, http://www.w3.org/RDF/, (1999).
18) W3C: RDF Schema, http://www.w3.org/TR/rdf-schema/, (2004).
19) W3C: SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/, (2006).
20) W3C: Named Graphs, http://www.w3.org/2004/03/trix/, (2004).
21) 橋田浩一: GDA:意味的修飾に基づく多用途の知的コンテンツ, 人工知能学会誌, Vol.13, No.4, pp528-535, (1998).
22) W3C: XML Schema, http://www.w3.org/XML/Schema, (2001).
23) Userland Software: XML-RPC, http://www.xmlrpc.com/, (1999).
24) HP: Jena - A Semantic Web Framework for Java, http://jena.sourceforge.net/, (2006).
25) MusicBrainz, http://www.musicbrainz.org/.
26) Good, M. MusicXML in Practice: Issues in Translation and Analysis, International Conference Musical Application using XML, pp. 47-54, (2002).
27) 後藤 真孝, 村岡 洋一: 音楽音響信号を対象としたビートトラッキングシステム-小節線の検出と打楽器音の有無に応じた音楽的知識の選択-, 情報処理学会 音楽情報科学研究会 研究報告, 97-MUS-21-8, Vol.97, No.67, (1997).
28) Hevner, K. : Experimental Studies of the Elements of Expression in Music, American Journal of Psychology, Vol. 48, pp. 246-268, (1936).
29) 後藤 真孝: SmartMusicKIOSK: サビ出し機能付き音楽試聴機, 情報処理学会論文誌 Vol.44, No.11, pp.2737-2747, (2003).
30) 梶 克彦, 平田圭二, 長尾 確: コミュニケーションメディアとしてのプレイリストを目指して, 情報科学技術レターズ, vol.4 pp. 115-118, (2005).
31) Daniel, E., Whitman, B., Berenzweig, A. and

Lawrence, S. : The Quest for Ground Truth in Musical Artist Similarity, Proc. of ISMIR, (2002).
32) Logan, B., Kositsky, A. and Moreno, P., Semantic Analysis of Song Lyrics, IEEE International Conference on Multimedia and Expo (ICME), (2004).
33) Salton, G. and Yang, C.S. :On the specification of term values in automatic indexing, Journal of Documentation, pp.29:351-372, (1973).
34) MoodLogic, http://www.moodlogic.com/.
35) Vinet, H., Herrera, P. and F. Pachet, F. : The CUIDADO Project, Proc. of ISMIR, (2002).

(平成 16 年 11 月 28 日受付)
(平成 17 年 2 月 4 日採録)



梶　克彦（学生会員）
2004 年名古屋大学工学研究科計算理工学専攻修士課程修了．現在，名古屋大学大学院情報科学研究科メディア科学専攻博士課程在学中



長尾　確
1987 年 東京工業大学 総合理工学研究科 システム科学専攻修士課程修了, 1987-1991 年 日本アイ・ビー・エム株式会社 東京基礎研究所, 1991-1999 年 株式会社ソニーコンピュータサイエンス研究所, 1996-1997 米国イリノイ大学アーバナ・シャンペーン校 客員研究員, 1999-2001 年 日本アイ・ビー・エム株式会社 東京基礎研究所, 2001-2002 年 名古屋大学 工学研究科 助教授, 2002 年- 現在 名古屋大学 情報メディア教育センター 教授