2004－HI－110　(8)
2004／9／10

異種ヒューマノイドロボット間でのコミュニケーションコンテンツの共有の実現

広瀬 健志郎† 佐竹 聡† 川島英之† 今井倫太 †‡
†慶應義塾大学理工学研究科 ‡科学技術振興機構さきがけプログラム

Abstract—本研究の目的は、コミュニケーションロボットのソフトウェアアーキテクチャに詳しくない一般ユーザが、ロボットを情報提供の手段として容易に利用できる手法を提案することである。ロボットの動作をプログラミングする手法は多数あるが、本研究の目的と照らし合わせると、それらは多種類のロボット間でのコンテンツ共有が不可、一般ユーザにとって使いにくい、センサ情報を反映したコンテンツ提示が不可のいずれかの問題を抱えている。よって本研究では、上記の 3 条件を満たすために、ロボットのコンテンツを直感的に記述できるエディタを開発と、そのコンテンツを多種類のヒューマノイドロボットで実行できるように、抽象的なモーションを具体的な関節情報に置き換える言語仕様を定義し、それらをロボットに実行させるシステムを開発した。

Realization of communication contents sharing among different types of humanoid robots.

Kenshiro Hirose † Satoru Satake† Hideyuki Kawashima† Michita Imai †‡
†Graduate School of Keio University ‡JST PRESTO

Abstract—Purpose of this research is to develop a system that enables general users to use communication robot as means of presenting information. There are several researches that deal with creation of contents, but all of them have some of following problems; Unable to share contents among different types of robots, the system is not easy enough for general users to use, can't use sensor data for presenting contents. Therefore in this research, we developed an editor that enables general users to make contents easily, and execution system to execute the contents.

1 Introduction

本研究の目的は、コミュニケーションロボットのソフトウェアアーキテクチャに詳しくない一般ユーザが、ロボットを情報提供の手段として容易に利用できる手法を提案することである。

近年多く開発されている様々なコミュニケーションロボットは、現実世界とインタラクションすることによる高い表現能力と、各種センサを用いることにより、キーボードやマウスを用いるよりも柔軟な情報入力方法を有する新しいメディアとして来たいされている。これらのロボットが普及すれば、ディスプレイやスピーカーだけでは表現しきれなかった広告、指導、案内等のコンテンツを十分に表現し、活躍できると思われる。

ロボットがそのように活躍するためには、ロボットのハードウェア面での充実に加え、ソフトウェア面でもロボットのコンテンツを作成する手法が充実しなければならない。そのようなロボットのコンテンツ作成方法は、以下の 3 点を満たす必要がある。1 点目は、多種類のロボット間でコンテンツが共有可能であることだ。コンテンツが共有不可能であると、一つのロボットで作成したコンテンツが他の種類のロボットで使えない。そうすると、ユーザはロボット毎にコンテンツ作成方法を覚えなければならず、またコンテンツの再利用もできないからだ。2 点目は、ロボットのソフトウェアやハードウェアに詳しくない一般ユーザにとって使いやすいということだ。関節の情報を具体的に数値等で入力しなければならない等、何をどこに記述すればロボットがどう動くのかが非熟練ユーザにとって感覚的に把握しにくいと、コンテンツを作成できるユーザが限られてしまう。3 点目は、ロボットがセンサ情報を用いて自発的にプレゼンテーションするコンテンツを選択できる記述が可能なことである。なぜならば、センサ情報を利用しなければロボットが現実世界を認識することができなく、コミュニケーションを通じてロボットが情報を提供することができないからである。しかし、これら 3 条件を満たす手法は現在開発されていない。

よって本研究では、上記の 3 条件を満たすために、コンテンツを多種類のヒューマノイドロボットで実行でき、抽象的なモーションを具体的な関節情報に置き換える言語仕様を定義し、それらをロボットに実行させるシステムを開発する。また、ロボットがセンサ情報を用いて自発的にプレゼンテーションするコンテンツを選択できる記述を直感的に記述可能なエディタを開発する。ユーザは、ロボットが人間に対して質問を投げかけてそれに人間が反応するロボット主導のコミュニケーションによりコンテンツをロボットにプレゼンテーションさせることができる。

以下、本稿ではまず 2 節にて本稿の背景として、従

来のコミュニケーションロボットの動作をプログラミングする手法について述べる。次に、3 節において本稿で提案するコンテンツ作成手法について述べる。そして??において今後の課題について述べ、最後に 5 節において結論を述べる。

2 背景

従来提案されてきたロボットのモーションを記述する仕様には以下が存在する。RoboML[1] の目的はインターネットを介したロボットの遠隔操作を記述することである。RoboML では、ロボットのハードウェアごとにモーションを記述できることができる。MotionML[2] の目的は、多種類のヒューマノイドでモーションデータを高い抽象度で共有することである。[2] では、XML の拡張言語である MotionML 自体の他にも、それを生成・管理・実行するシステムを提案し、様々なヒューマノイド間でモーションファイルの共有を実現している。シナリオエディタ [3] は、ヒューマノイドロボット Robovie-R と Robovie-M 附属のモーションエディタであり、Robovie-R と Robovie-M のモーションの作成を目的としている。モーションの作成・再生機能の他に、複数の動作を順番に自動再生する機能、移動中に障害物を避ける反射動作機能、各センサ出力の即時表示機能を持っている。

しかし、これら 3 つの手法は前述した 3 条件を満たすものではない。RoboML はハードウェアを直接操作するレベルでしか記述できなく、高い抽象度で動作を記述できないことから第 2 の条件を満たせない。MotionML は、センサを搭載したヒューマノイドを想定してないため、ヒューマノイドの自発的コンテンツ選択を実現することが不可能であるので、第 3 の条件を満たせない。シナリオエディタは、アプリケーションがロボットに依存した形式になっており、違う種類のロボットでシナリオエディタにより作成したのファイルを共有することが難しいため第 1 の条件を満たせない。また、センサに対してとれる反応も障害物を避けるといった単独の動作なので、センサ情報に応じてプレゼンテーションするコンテンツを選択させることができないため第 3 の条件も満たせない。

3 コンテンツ作成・実行システム

本節では、本稿で提案するコンテンツ作成・実行システムについて述べる。まず 3.1 節においてコンテンツ作成・実行システムのコンセプトについて述べる。次に、3.2 節でユーザがロボットのコンテンツ作成に使用するコンテンツエディタの仕様を説明する。次に、3.3 節においてコンテンツエディタが出力するロボットコンテンツファイル (RC ファイル) について述べる。そして次に、3.4 節において抽象的なモーションを具体的な関節情報に置き換え、ヒューマノイドロボット間で共有するための言語仕様 Humanoid Motion Description Language(HMDL) について述べる。最後に、3.5 節においてコンテンツエディタにより作成したコンテンツをロボットにプレゼンテーションさせるシステム (RC エグゼキュータ) について述べる。

3.1 コンテンツ作成・実行システムのコンセプト

本稿では 3 条件を満たすコンテンツ作成手法を開発した。この手法では、ユーザ、システム開発者、ロボット開発者の 3 者がそれぞれ分担作業することでコンテンツを実行できる。

まず、システム開発者である本稿の筆者がロボットのモーションとセンサイベントを定義する。モーションでは、どの関節がいつ・どの方向に・どれだけ曲がるのかを定義する。例えば、「右を指す」というモーションは肩の水平方向の関節を 1 秒で 90 度曲げる、というものである。モーションは 3.4 節で詳しく述べる HMDL により記述される。センサイベントでは、どのイベントがどのセンサに対応しているのかを定義する。例えば、「右に人が近づく」というイベントは横方向の距離センサが 1m 以下になった場合、というものである。これらのモーション、センサイベントを各数十個作成する。

ロボット作成者は、システム開発者が定義したモーションやセンサイベントをそのロボットで実際にモーターやセンサデバイスにアクセスする方法、つまりデバイスドライバ、を記述する。個々のロボットによって「1 秒で肘の関節を 90 °曲げる」ためにどのモーターにどれだけ電圧をかけるかは各ロボットによって変わるし、距離センサや接触センサもロボットによって若干違うためである。

ユーザは、システム開発者が作成したロボットのモーションとセンサイベントを利用してロボットのモーションと発話の同期であるコンテンツを作成する。基本的にはシステム開発者が作成したモーションとセンサイベントの切り貼りによりコンテンツを作成するが、モーションの速度、繰り返しの有無、モーションの大きさを調整することができる。

3.2 コンテンツエディタ

コンテンツエディタは、ロボットの動作、モーション、センサイベントの同期を記述するためのアプリケーションである。一般ユーザが扱いやすいように、画面上に現れる CG ロボットで具体的なモーションを確認しながら、ロボットのモーションやセンサイベントを抽象度の高い表現方法で記述できるようになる。図 1 にコンテンツエディタの画面のサンプルを示した。

図 1 の各部分について説明する。

1. ロボットタイプセレクタ — 車輪型ロボットか二足歩行かを選択。マウスを使うことで横方向・縦方向に CG ロボットを回転させられる。
2. ロボットディスプレイ — CG キャラクタのロボットが表示される。
3. モーションセレクタ — スケジュールに登録するモーションを選択する部分。

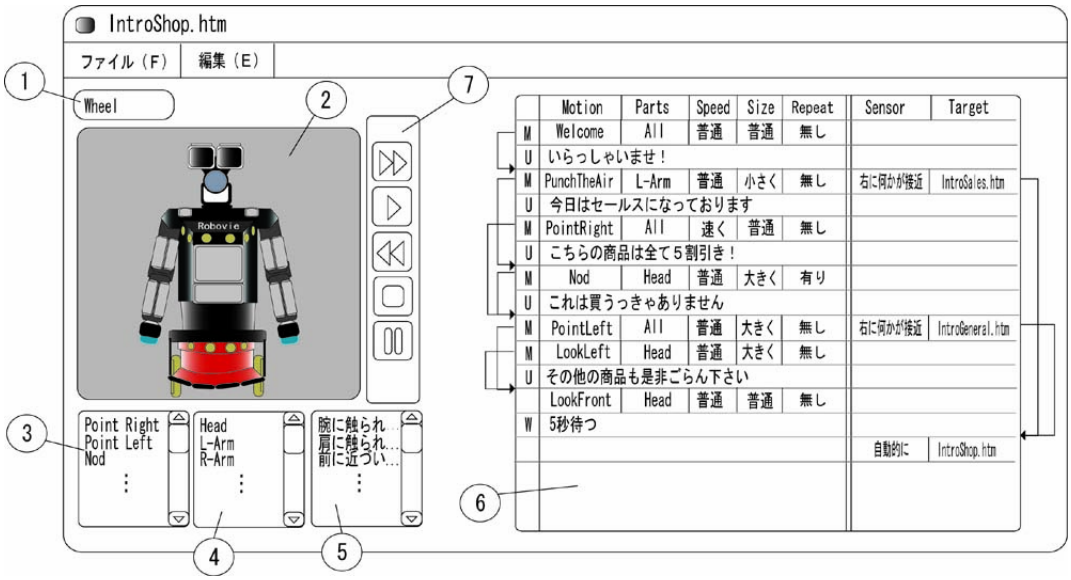

図 1: コンテンツエディタ

4. パーツセレクタ — モーションの特定の関節部分だけを使いたい際に、使用する関節部分を選択する。リストされているパーツをダブルクリックするとそのパーツの使用・不使用を選択できる。

5. センサセレクタ — スケジュールに登録するセンサの種類を選択する。

6. スケジュールディスプレイ — 左側にロボットの発話、モーションが書いてあり、右側にセンサイベントが時系列に並んでいる。ユーザはここでモーション、発話、センサの同期を記述する。スケジュールディスプレイの左右にある矢印は、そのモーションやセンサイベントが有効な期間を示している。図 1 だと、最初の Welcome というモーションは「いらっしゃいませ！」という発話が終わるまで続けられる。

7. シミュレーションコントローラ — 作成した RC ファイルがどのようにプレゼンテーションされるかをシミュレーションするためのコントローラ。再生、早送り、巻戻し、一時停止、停止ができる。

ユーザがこのエディタを用いてコンテンツを作成する手順は以下のようになる。

ロボットタイプの決定　ロボットタイプセレクタにより、使用するロボットが車輪型か二足歩行型かを決定する。

ファイル読み込み　ファイルからプレゼンテーション内容を読み込みたい場合、HTML ファイルやプレインテキスト等のテキストファイルを読み込む。

スケジュールの作成　スケジュールの作成では、発話とモーションのスケジュールを決定するモーションスケジュールの作成と、センサ情報をもとに他のコンテンツへリンクするタイミングを作成するセンサスケジュールの作成がある。

1. モーションスケジュールの作成 — ここでは、モーションと発話がどのような順番で行われるのかを指定する。スケジュールディスプレイの左半分にモーションスケジュールは作成することができる。発話内容を配置したい場合は直接スケジュールディスプレイに書き込む。以下にその具体的な手順を示す。

   (a) モーションセレクタの中にあるモーションをクリックする。するとロボットディスプレイの中のロボットがそのモーションを実際に行うので、それを見て具体的にどのようなモーションなのか確認する。

   (b) ドラッグ&ドロップによってスケジュールディスプレイの任意の行にモーションを配置する。

   (c) モーションがスケジュールのどの範囲まで適用されるのかを指定する。

   (d) モーションについて、速度を「速い」、「普通」、「遅い」の 3 値か、デフォルトの何%の速さかを自然数で、記入する。また、モーションの大きさも、「大きい」、「普通」、「小さい」の 3 値か、デフォルトの何%の大きさで実行するかを自然数で指定する。

   (e) そのモーションを指定した期間繰り返し行うか、1 回のみ行うかを指定する。

2. センサスケジュールの作成 — ここでは、センサ情報をもとにコンテンツにリンクするタイミングを記述する。センサスケジュールはスケジュールディスプレイの右半分に記述することができる。

   (a) センサセレクタの中にあるセンサイベントをドラッグ&ドロップによってスケジュールディスプレイの任意の行に配置する。

   (b) センサイベントが有効な期間を指定する。

(c) センサイベントが発生した際にどのコンテンツにリンクするのかを相対パスか URI により指定する。

シミュレーション 一連のスケジュールができあがったら、それをどのようにロボットがプレゼンテーションするのか確認するために、シミュレーションを行う。シミュレーションを行う際は、シミュレーションコントローラを用いて行う。シミュレーション中のエディタの様子を図 2 に示した。

シミュレーション中は、ロボットのモーションにあわせて、ロボットの発話内容がロボットディスプレイの下に吹き出しの形で表示される。また、現在スケジュールのどこをプレゼンテーションしているのかを、行をハイライトすることで知らせてくれる。

ファイル出力 シミュレーションが終了したら、スケジュールをファイルに出力する。ファイルはロボットコンテンツファイル (RC ファイル) というテキストファイルになっており、発話以外の情報はタグにより記述されている。RC ファイルの例を以下に示す。これは図 1 の最初から「こちらの商品は全て 5 割り引き」の発話までを RC ファイルにしたものである。

```
(motion-begin: title=Welcome parts=All
speed=normal size=normal repeat=none)
いらっしゃいませ！
(motion-end: title=Welcome)
(motion-begin: title=PunchTheAir
parts=L-Arm speed=normal size=small
repeat=none)
(sensor-begin: trig=SmgOnRight
IntroSales.htm)
今日はセールスになっております
(motion-begin: title=PointRight
parts=All speed=fast size=normal repeat=none)
こちらの商品は全て 5 割り引き！
(motion-end: title=PunchTheAir)
```

3.3 RC ファイル

RC ファイルはタグとテキストから構成されており、タグがセンサイベントとモーションに関する記述を示しており、テキスト部分が発話を示している。ここでタグとは、( ) でくくられているテキスト部分のことで、その中の：の前までをタグネーム、それ以後に付加されている情報をアトリビュートと呼ぶ。以下に RC ファイルで記述できるタグの種類と効果を示した。

motion-begin,motion-end

motion-begin タグはモーションの開始を示すタグであり、以下のアトリビュートを持つ。

title — 使用するモーションファイルの名前。
speed — そのモーションをデフォルトの何倍の速度で実行するかを決定する。
repeat — モーションを繰り返すか否かをを指定する。
part — モーションファイルのうちどの関節部分を利用するかを指定する。r-shoulder(右肩)、r-f-arm(右前腕)等、ある関節よりも身体から外側の部分のモーションを利用することができる。
size — そのモーションをデフォルトの何倍の大きさで関節を動かすかを指定する。
motion-end タグはモーションの終了を示し、title アトリビュートのみを持つ。

control

control タグはモーションが行われている最中に、そのモーションの詳細を変更するタグである。control には以下のアトリビュートがある。
name — 変更したいモーションの名前。
speed — モーションの実行スピードを指定する。デフォルトの何%のスピードかが記述されており、0 の場合、ロボットはその姿勢のまま静止する。
size — motion-begin タグとの size アトリビュートと同じ。
part — motion-begin タグの part アトリビュートと同じ。

sensor-begin, sensor-end

sensor-begin は、どのセンサ情報を得た時にどんなコンテンツをロボットがプレゼンテーションするか指定する。sensor-begin タグは以下のアトリビュートを持つ。
trig — どのセンサ情報がリンクの引金になるのかを指定する。終了タグが読まれるまでこのトリガは有効となる。以前にも同じトリガを別の sensor-begin タグで使用していた場合は上書きとなる。
href — リンク先のコンテンツの相対パス、又は URI を指定する。

sensor-end は trig アトリビュートのみを持ち、trig のセンサイベントを無効にする。

wait

wait タグは、一定の時間だけ RC エグゼキュータを止めることができる。wait エレメントは、time アトリビュートを持つことができる。RC エグゼキュータが止まる時間を秒で指定することができる。

3.4 HMDL の言語仕様

HMDL では、RC ファイル中の motion-begin や motion-end の title アトリビュートで指定されたモーションが具体的にどのような動作を示すのかを、ロボットの関節ごとに表現した言語仕様であり、XML の拡張言語である。HMDL はロボットの身体が階層構造で示されることに注目し、各関節を入れ子構造にして記述し、各関節が曲がる角度、方向、曲がり始めてから終わるまでの時間を指定する。

hmdl エレメント

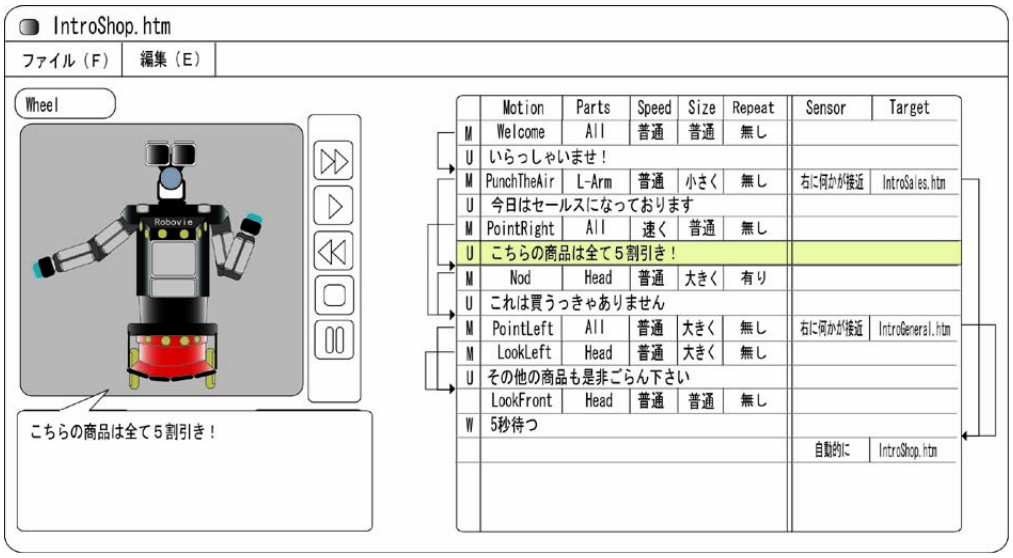

図 2: シミュレーション中のコンテンツエディタ

hmdl エレメントは以下のアトリビュートを持つことができる。

type — そのロボットが車輪型 (wheel) か二足歩行型 (bipedal) かを示す。

hmdl エレメントは body を子として持つことができる。

body エレメント

body エレメントは phase を子として持つことができる。

phase エレメント

phase エレメントは、モーションの 1 つのセグメントを示すものである。phase エレメントは以下のアトリビュートを持つ。

duration — そのモーションセグメントが何秒間で終了するかを指定する。整数値をとることができる。

phase エレメントは子として関節エレメント群の最上位エレメントを内容として持つことができる。

関節エレメント群

関節エレメント群とは、図 3 のようなロボットの身体が階層構造になっている点に注目して、ヒューマノイドロボットの関節それぞれをタグで表したものの総称である。例えば、<left-arm> エレメントの中には <left-shoulder>、<left-elbow>、<left-wrist> があり、<left-shoulder> エレメントの中には <left-shoulder-horizontal>、<left-shoulder-vertical>、<left-shoulder-rotate> がある。

下半身部分の HMDL については、二足歩行型ロボットと車輪型ロボットの 2 種類をそれぞれ定義する。

関節エレメント群はそれぞれ position を子として持つことができる。

position エレメント

position エレメントは関節をどちらにどれだけ曲げるのかを指定するエレメントである。position エレメントは以下のアトリビュートを持つことができる。

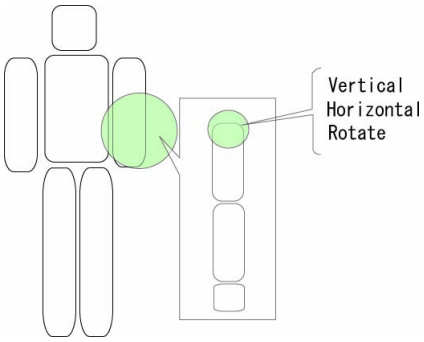

図 3: ロボットの身体の階層構造

direction — 曲がる方向。forward か back のどちらか。

angle — 曲がる角度。関節によって可能な指定範囲は異なる。

3.5 RC エグゼキュータ

ここでは、RC ファイルを実行するシステムに関して述べる。RC ファイルを実行するシステムは、RC エグゼキュータ、HMDL エンジン、ロボット API の 3 つから成る。システムの概要を図 4 に示した。これらのモジュールは全て C/C++で実装されている。

RC エグゼキュータは RC ファイルを構文解析し、ロボットの API に直接、発話を要求したりセンサ情報を要求する他、モーションファイルを HMDL エンジンに渡して具体的な関節情報に置き換えロボットに実行させるように要求する。また、どのモーションやセンサイベントが有効かを管理する。

HMDL エンジンは RC エグゼキュータからモーションファイルを受取り、それを具体的な関節情報にしてロボットに実行させる。

ロボット API は、HMDL エンジンから受け取った具体的な関節情報をモーターの電圧値に変換、RC エグゼキュータから受け取った文字列をロボットの発話に

変換、RC エグゼキュータから要求されたセンサ情報をデバイスにアクセスし調べるという 3 つのことを行う。

RC エグゼキュータは、RC ファイルを実行するアプリケーションである。RC エグゼキュータは構文解析機、モーション制御モジュール、リンク制御モジュールから成っており、これらのモジュールは独立して動作する。スケジュールエンジンの概要を図 4 に示した。

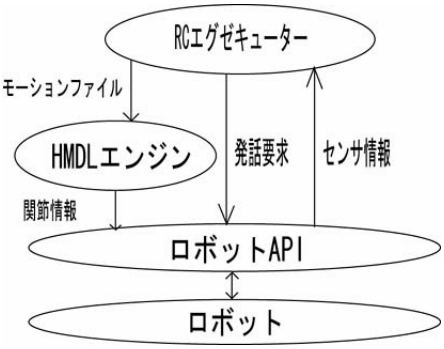

図 4: RC ファイル実行システムの概要

4 今後の課題

本稿ではコミュニケーションロボットのコンテンツを共有するための方法として、モーションやセンサイベントを共有できる形であらかじめ定義しておくという方法をとったが、現在のままでは自由度の違うロボット間で共有が難しくなってしまうモーションが存在する。例えば、「右上を指す」といったモーションの場合に、図 5 にあるように指してしまうと、肘から先の関節がないロボットでは本来の意味が失われてしまう。ここにはさらに一段階上の抽象化が必要であると考えられる。また、センサイベントに関しても、現在はロボットに不足しているセンサについては後付けしなければならないので、センサイベントの抽象化も進める必要がある。

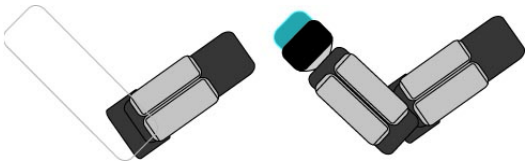

図 5: 失われてしまうモーションの意味

5 結論

本研究では、コミュニケーションロボットのソフトウェアアーキテクチャに詳しくない一般ユーザが、ロボットを情報提供の手段として容易に利用できるコンテンツ作成手法を開発した。そのような手法には 3 つの条件が必要であるが、コンテンツが異種間のロボットでも共有可能でなくてはならないという第 1 の条件は、ロボット間で共有できるモーションやセンサイベントをあらかじめ定義することで解決した。第 2 の条件はである一般ユーザに対しての使いやすさは、既存のモーションやセンサイベントを切り貼りするだけでコンテンツを作成できるエディタを開発することにより解決した。第 3 の条件であるロボットによる自発的コンテンツ選択は、センサイベントによりロボットがコンテンツを選択する記述をエディタ内で記述可能にしたことにより解決した。

参考文献

[1] Maxim Makatchev, S.K.Tso, “Human-Robot Interface Using Agents Communicating in an XML-Based Markup Laguage”, Proceedings of the 2000 IEEE International Workshop in Robot and Human Interactive Communication, September 27-29

[2] Kitagishi et al., “Development of Motion Data Description Language for Robots Based on eXtensible Markup Language -Realization of Better Understanding and Communication via Networks-”, IROS2002

[3] ATR 2003
http://www.atr.jp/html/topics/press_031028_j.html