平成23年度 卒業論文

論文題目

著作権管理のためのBee-gentによる仲介システム

神奈川大学 工学部 電子情報フロンティア学科

学籍番号 200702972

山口　誉央

指導担当者　　木下　宏揚　教授

目 次

図 目 次

表 目 次

第1章

序論

パソコンとインターネットの普及に伴い、昨今では数多くのディジタルコンテンツが数多くの製作者によって作成されている。製作者は企業などの著名な製作者たちに限らず、一個人が製作したコンテンツも数多く存在している。そんな時代の中、それらのコンテンツを配信、利用するにあたり、著作権の保護や利用の制限をどう取り扱うか、どのように制御していくのかが、インターネットの普及から十数年経った今でも問題となっている。

著作権という権利の歴史は古く、18 世紀初頭、今からおよそ 300 年前から続いている。最初の著作権は印刷技術の向上により、出版物を容易く複製することが出来るようになったために考え出されたものであった。今日海賊版と呼ばれている違法複製品は当時から存在していた。しかし現在ほど強く意識されていた事はなかったと思われる。何故なら、従来では作品を製作し、公衆に提供するためには多大な労力が必要であった。また、印刷技術・印刷機構を持たない一般人は業者が提供するものをただ享受する存在であったし、同じ理由で違法複製品を製作でき、公衆に提供することが出来るのも業者か、それに類するものに限られるため、国の法律がそれらを取り締まる事も容易だった。[2]

しかしインターネット時代の到来と共に、従来のようにスムーズに問題を解決する事が非常に難しくなってしまった。ディジタル化されたデータは、パソコンひとつで劣化しない複製品を製作できてしまうし、インターネットを介してディジタルデータを世界中に向けて発信するなど造作もない。何より、今まで一般人と呼ばれていたただ享受するだけだった者たちの中から、多数の製作者が出てきてしまっている。インターネットにより、人に認知される作品とその製作者が大幅に増えたため、もはや従来の法律では勝手が追いつかなくなってしまっているのだ。[5]

そんなインターネット時代の都合に合わせて、今では製作者の意図に従えばコンテンツの二次利用が利用者に一任されるようなものもある。[8]

しかし、それでも製作者と利用者との間には長い距離があり、利用者が製作者の意図に従ったつもりでも、製作者側がそれを認めていなかったなどの事例も多い。

そこで、利用者と製作者の意思疎通を補助するためのツールとして、東芝研究開発センター知識メディアラボラトリーが開発した Bee-gent に着目する。Bee-gent はコミュニケーションのエージェント化という新しい概念を実現するために作成されたものである。Bee-gent はコンテンツをエージェント化するエージェントラッパーと、エージェントラッパー間を仲介するエージェントラッパーで構築されている。コンテンツをエージェント化することで既存コンテンツを活かして柔軟で開放的な分散システムを構築し、コミュニケーションを円滑に進めることを目的として作られた。エージェント化したコンテンツに製作者の意図を利用者に伝える役割を持たせ、仲介エージェントによりそれを利用者側のエージェントに送ることを目的とする。

第2章

基礎知識

2.1　ディジタルコンテンツの問題点

ディジタルコンテンツの著作物に関する特徴は

- 複製が容易
- 流通が容易
- 改変及び加工が容易

というような特徴を持つ。[1]

一般的に著作権が管理されている状態とは、広く一般に配布するコンテンツに対して、以下のような条件を満たしている状態の事を指す。

- 適正なｎ次利用が促進されること
- 不正なコピー行為などに対する防衛策が講じられていること

インターネットを介してコンテンツを配布する際、そのコンテンツがどこに存在するのかを効率よく検索できるシステムが重要となる。その手段の一つとして、コンテンツのオントロジーとしてのメタデータを、情報の分類・整理・検索に利用するというものがある。このようなシステムにおいて、ディジタルの特徴を活かし、かつコンテンツの利便性を向上させるためには

- 流通後に利用条件やコンテンツの変更が可能であること
- 適正な二次利用が可能であること

上記の条件が必要である。

2.2 コンテンツの保護

コンテンツの保護という点では、著作権をはじめとする権利関係とデータの暗号化等の技術関係に分けられる。しかし、分けられるといってもこの二つは別々に考えるべきものではない。そもそもモノを複製するという技術は、優れた著作物をより普及するために作られたものであり、その歴史は15世紀のドイツにまで遡る。ドイツのグーテンベルグが活版印刷機を発明したことにより聖書がヨーロッパ社会に普及したのだ。今でも世界中に老若男女のキリスト教徒が大勢居る事を考えれば、この最初の複製技術が人間社会に与えた影響が膨大である事は想像に難くない。しかし、今度は印刷技術がすすんだために出版物の海賊版が出回るようになってしまい、これを規制する試みが始まった。イギリスでは1710年に版権保護の法律が定められ、その後デンマークやアメリカ、フランスなどでも出版に関する法律が制定された。しかしその後しばらくの間、著作権は国際的な法律とは言えなかったため、その状況を改善するため国際的な著作権保護の運動が活発となり、1886年にスイスのベルンにて、各国の代表が集まり、ヨーロッパ各国が参加する国際的な著作権保護条約が結ばれた。つまり、著作権保護の法律とは、技術が進んだために制定されたものなのだ。逆に暗号化などの技術は著作権を保護するために進んだ技術である。著作権の保護という課題において、法律と技術は切っても切れない関係である事は、実は100年以上も前から続いていたのである。[3]

2.3 DRM

DRMとはディジタル著作権管理（Digital Rights Management）の訳であり、ディジタルデータとして表現されたコンテンツの著作権を保護し、その利用や複製を制御・制限する技術の総称である。今日のコンテンツ保護の多くは暗号化である。権利取得者以外はコンテンツが利用できないというのがコンテンツ保護の本来の姿である。しかし現状は、企業どころかその企業が開発するコンテンツ毎に用意し利用するか、或いは技術の提供を受けて独自開発するかであり、一本化されるまでには至っていない。著名なものではMicrosoftのWindowsMediaDRMが存在し、現在もバージョンアップを続けている。[6]

ただしDRMへの批判意見も多い。代表的なものとして

- 恒久的な再生が保障されていない
- 消費者の権利に対する制限
- 特定環境への依存

などの批判がある。上記の通り、DRM技術のほとんどが特定のメーカーによって一意的に定められ、その技術的な詳細が一般には公開されていないことから、そのメーカーやサービスが活動を停止した際に、購入したコンテンツが将来にわたっても利用可能なのかが必ずしも保障されていない。またメーカー側が活動を停止せずとも、再生機器を買い替えた場合にデータの移行が出来ず購入したコンテンツが利用できなくなる等の可能性もある。そのうえDRMはその技術的な特性のため、違法な複製以外の利用著作権により認められている

- 私的複製
- 抜粋
- 編集

などの行為も制限してしまう場合も多い。そのためDRMは購入したコンテンツを自由に使う消費者の権利を奪っているということで、DRMをDigital Restrictions Management、ディジタル諸制限管理と呼ぶべきだという意見まであがってしまっている。

2.3.1 Description と enforcement

DRM の中核を為すシステム、「Rights Description」と「Rights Enforcement」。訳すと前者は権利を記述するもの。つまり DRM を記述・設定するための技術の総称である。対して後者は権利を施行するもの。すなわち実際に権利を施行するシステムを指す。[11]

Description により記述された内容や条件などを Enforcement が読み取り、施行するのが今日の DRM の主流である。例をあげると、コンテンツを利用するためのライセンスキーがある。これは Description によってライセンスキーに記述された情報を Enforcement が読み取り、ライセンスキーに記述された条件や契約に違反しない範囲でのコンテンツの利用を許可する、というものである。ライセンスキーに限った話ではないが、ある法則により Description は記述を行い、それと同じ法則で Enforcement は情報を読み取るため、この二つは同じ法則に完全に依存している事が多い。そのため法則が違えば Enforcement は記述された情報を読み取る事が出来ない。この法則こそが「暗号」であり、これによりコンテンツを保護している。ただし、この法則は前述の通り各製品により完全に独立しているため、良く言えば暗号性が高いと言えるが、言い換えれば互換性が一切無いとも言える。これが前述の DRM の問題点に直結している。

2.3.2 クリエイティブ・コモンズ・ライセンス

クリエイティブ・コモンズ・ライセンスとは、インターネット時代の新しい形の著作権ルールの普及を目指し活動す国際的非営利組織が提案する、様々な作品の作者が「自分が定めた条件を守れば作品を自由に利用して良い」という意思表示のための新しい著作権管理の形である。[8]

人が作り出した全ての作品は著作権で守られているか、そうでないかに分けられる。ただし、上記の著作権は複製権や上映権と言った著作権が有する権利の全てを指す。クリエイティブ・コモンズ・ライセンスはこの二つの状態の間の選択肢と言える。著作物の中間領域を定義し、「Some rights reserved」、つまり限定された権利のみを主張するライセンス形式を提案している。作品の利用（再配布や改変作品の公開及び実演など）のための条件４つを定めており、

- 表示：原作者のクレジット（氏名、作品タイトルとURL）を表示すること

を基本として、それに以下の条件を１つないし２つ選ぶ。

- 非営利：営利目的での利用をしないこと
- 改変禁止：元の作品を改変しないこと
- 継承：元の作品と同じ組み合わせのライセンスで公開すること

クリエイティブ・コモンズは「このライセンスはDRMか？」という質問に対し

- 「ライセンスのEnforcement＝執行という役割を、法律や社会規範、そして参加者の良心に任せている」
- 「このツールは規制の道具としてではなく、補助的な情報として機能し、著作権利者が自分の著作物に対する義務と自由について他の人々に教えてもらい、インターネット上でクリエイティブな再利用が出来る場所を見つけたい」

これらを方針にして活動していると回答している。[10]

クリエイティブ・コモンズは３つの要素により、その効果を保証しようとしている。第一に法律に詳しくない人でもライセンス内容が理解できるよう簡潔な説明文が書かれた「コモンズ証」第二に同じ内容を法律の専門家が読むために法的に記述した「利用許諾」最後にRDF構文に基づいて記述されたメタデータを用いて、人だ

けではなく検索エンジンやプログラムが利用条件を理解するためのコードを付与することで、他のユーザーに製作者の作品が正しく検索されやすくしている。[9]

2.4 ソフトウェアエージェント

ソフトウェアエージェント（以下エージェント）とはユーザーや他のソフトウェアとの仲介役として動作するシステム。人間が本来やるべき情報アクセスや処理をプログラムによって代行するような技術の総称でもある。[7]

現在はユーザーの行動を補佐したり、より効率的に情報を引き出すなどに留まっているが、究極的な形として、人間に取って代われるほどの知能を持たせるという目標があるシステムでもある。エージェントとして機能しているシステムには現在明確な定義がなく、かなり多様性がある。以下はその一例である。

- エージェントが持つ、他のエージェントとの通信機能を使って情報を交換したり、タスクの遂行を要求したりする同期・非同期の通信機能（コミュニケーション）
- エージェント自身が、活動する情報処理環境内で移動出来る移動機能（モビリティ）
- 推論やプランニングなど、問題解決に関する知的能力（インテリジェンス）
- そのシステムは要求されて実行するのではなく、常に起動された状態で、何らかの行動を起こす時期をシステム自体が判断する。（パーシスタンス）
- 実行すべきタスクの選択、優先順位の決定、意思決定を人間の手助けなしで行う機能を持つ。（オートノミー）

2.5 Bee-gent

東芝研究開発センター知識メディアラボラトリーが開発したエージェントフレームワーク。正式名称は Bonding and Encapsulation Enhancement aGENT. 既存アプリケーションをエージェント化するエージェントラッパーと、アプリケーション間の連携手続を組み込む仲介エージェントの 2 種類で構成される。仲介エージェントはアプリケーションの存在する場所を移動し、エージェントラッパーと情報を交換（対話）しながら連携手続を実現する。エージェントラッパーはアプリケーションの状態を管理し、必要に応じてアプリケーション処理を起動することで仲介エージェントからの要求に応える。[16]

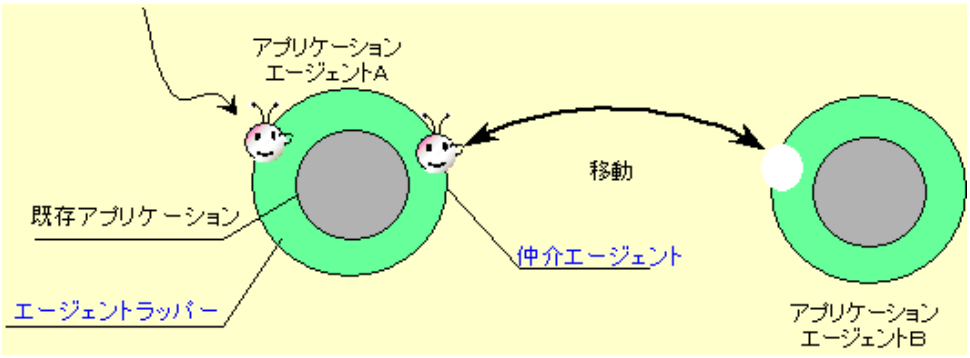

図 2.1 Bee-gent の動作イメージ

Bee-gent には以下のような特徴がある。

- 仲介エージェントがアプリケーション間の連携手続きを一元管理

連携手続きを分割・分散せずにすむため整合性確保が容易となり、システム設計が単純化される。連携手続きそのものがカプセル化されるため、システム構成や連携手続きの追加や変更が容易に可能。

- 仲介エージェントが連携手続き／データ／状態を持って自ら移動

メッセージで連携する場合に比べ通信頻度が減り、また、エージェント送出後は通信回線を切断できるなど、ネットワーク負荷が減少する。移動先で相手とローカルに情報交換を行うため、効率的な処理が実現できる。

- エージェントラッパーが共通インタフェースを提供

相互運用性が高くなり、開放的なシステム構築が容易となる。連携先のアプリケーションの利用手続きに関する知識が不要となり、大規模システム開発が可能となる。

- エージェントラッパーによるアプリケーションのエージェント化と ACL による対話を実現

仲介エージェントからの要求に自律的に対応することにより、高度な情報保護や処理の優先処理などが可能。エージェント化されたアプリケーションと仲介エージェントはエージェント間通信言語 ACL(Agent Communication Language) を用いて対話することにより、状況変化に対応した柔軟なやりとりが出来る。

- インターネットや Web Top コンピューティングにも親和性の高い情報交換方式の採用

仲介エージェントの移動や通信には、HTTP(Hyper Text Transfer Protocol) を採用。仲介エージェントとエージェントラッパー間の対話言語である ACL の表現形式として今後普及が期待されている XML(eXtensible Markup Language) を採用している。

2.5.1 エージェント間通信言語

ACL(Agent Communication Language) は発話行為理論 (Speech-Act-Theory) に基づいたエージェント間通信言語。対話参加者の間で会話を進める際に行う意味的な行為をパフォーマティブ (performative, 通信行為) と言う。パフォーマティブをメッセージとして相手に送信することによって，ある行為の発生が意図される。例えば，request というパフォーマティブは，送信側が受信側に対して，何らかの行為の実施を要求することを意味する。一方、受信側にとっても会話の内容がどのような種類の応答を期待しているかを判断することが可能。Bee-gent が標準でサポートしているパフォーマティブは以下の通り。

- request ：送信側が受信側に対して，何らかの動作の実行を要求する.
- query ：送信側が知らない情報について，受信側に問い合わせる.
- inform ：送信側が知っている情報を通知する.
- cfp ：ある動作の実行の申し込みを呼びかける.
- propose ：ある特定の事前条件の下で特定の動作を (送信側が) 実行することを提案する.
- accept-proposal ：ある動作を実行するために，先だって申し出のあった提案を受理する.
- reject-proposal ：先だって申し出のあった提案を却下する.
- agree ：何らかの動作を実行することに同意する.
- refuse ：ある動作の実行が可能ではないために拒否する.
- failure ：ある動作の実行を試みたが，何らかの理由により失敗に終わったことを通知する.
- not-understood ：ある行為を受信したが理解できなかったことを，その行為の送信側に通知する.

2.6 関連研究

2.6.1 スマーティブプロジェクト

コンテンツに提供者や利用者の要求をポリシーとして埋め込み、自ら考えるコンテンツを実現する技術を開発する取り組み。国立情報研究所の本位田真一氏により研究されている。[13]

安全と自由のバランスがとれたコンテンツの流通・活用を可能とするため、提供者・利用者双方のポリシーをすり合わせる。そのためのエージェントフレームワークを使うことで、適切な流通システムの構築を行う活動をしている。[12]

2.6.2 XML

文書やデータの意味や構造などを記述するためのマークアップ言語の一つ。マークアップ言語とはタグと呼ばれる特定の文字列で、地の分に情報の意味や構造、装飾などを埋め込んでいくための言語のことで、XML はユーザー自身が自由にタグを指定できるため、マークアップ言語を作成するためのメタ言語ともいわれる。また、メタ言語とは言語を記述するための言語の事を指す。[14]

XML により、統一的な記述法を用いつつ、独自の意味や構造をもったマークアップ言語を作成できるため、ソフトウェア間の通信・情報交換に用いるデータ形式、様々な種類のデータを保存するためのファイルフォーマットの定義などに使われている。XML を様々な場面で利用するための関連技術の規格も数多く存在している。中でも有名なのが、文書内に埋め込まれた他の文書や画像などの位置情報をハイパーリンクを用いて複数の文書や関連する画像などのオブジェクトを関連付けたハイパーテキストというシステムである。このハイパーテキストの代表格は WWW であり、Web ブラウザで文書を表示し、リンクのある場所をマウスでクリックすれば XML で記述され関連付けられたリンク先にジャンプする、という仕組みになっている。[4]

2.6.3 RDF

Resource Description Framework 通称 RDF とは Web 上にあるリソースを記述するために統一された枠組みであり、W3C（World Wide Web Consortium）により規格化されている。RDF は特にメタデータについて記述する事を目的としており、WWW の利便性の向上を実現するための技術的な構成要素の一つとなっている。RDF は主語となるリソースを述語と目的語によって表現する。RDF のモデルは主語と目的語をノードとし、述語をこれらを結ぶアークとする有向ラベル付きグラフとして表現される。この時、目的語がリソースである場合はその目的語を主語にした別の分が続くという連鎖型のモデルをとることができ、グラフとしてはノード、アーク、ノード、アーク、… という連鎖が示される。また、同じ主語に対する複数の文を表現するため、一つのノードから複数のアークが伸びる場合もある。前述のクリエイティブ・コモンズはこの構文に基づきメタデータの記述を行っている。[15]

第3章

提案システム

3.1　行った研究

● Bee-gent による利用者と製作者の仲介利用者による作品の二次利用の要求を製作者に伝えるためのプログラムを Bee-gent によって作成し、両者の意思疎通を補助するエージェントシステムを作成しようと試みた。次に簡単な Bee-gent のエージェントラッパーと仲介エージェントの例を表した。

3.1.1 Bee-gent のプログラム例

● HelloWorld プログラムのエージェントラッパー例
エージェントラッパー AW1 が仲介エージェント Bee1 を作成し、その Bee1 からの要求により HelloWorld と発言する

```
 1  import java.io.*;
 2  import java.net.*;
 3  import java.util.*;
 4
 5  import com.toshiba.beegent.*;
 6  import com.toshiba.beegent.util.*;
 7  import com.toshiba.beegent.xml.*;
 8
 9  public class AW1 extends AgentWrapper{
10
11    public static void main(String[] argv) throws Exception{
12
13      AW1 aw1 = new AW1();
14      aw1.addIPStates(new AW1IPState1());
15
16      aw1.startIP();
17
18    }// main
19
20  }// AW1
21
22  class AW1IPState1 extends AwrIPState{
23
24    AW1IPState1(){
25      setPrecond("INIT");
26      setPostcond("END");
27    }
28
```

1 行目から 20 行目までがエージェントラッパーのクラス定義である。9 行目でエージェントラッパーの名前を AW1 として定義している。Bee-gent はエージェントラッパーの状態をクラスとして定義し、14 行目でそのインスタンスをメソッド addIPStates() を用いて登録している。そして、14 行目で登録した AW1IPState1 のクラスを 22 行目から定義している。25 行目で事前条件を”INIT”、26 行目で事後条件を”END”と定義している。

```
29    public void action(){
30
31      System.out.println("Talking with Bee1...");
32
33      // Create Bee
34      try{
35        createBee("Bee1");
36      }catch(Exception e){
37        System.out.println("Failed to create Bee.");
38        return;
39      }
40
41      // Receive XML/ACL
42      while (waitXML(0)){
43        XmlAcl xa = getXML();
44        String perf = xa.getTag2Value("performative");
45
46        if (perf.equals("request")){
47          String action = xa.getTag2Value("action");
48          String args   = xa.getTag2Value("args");
49
50          if (action.equals("say")){
51            System.out.println(args);
52            break;
53          }
54        }
55      }
56    }// action
57  }// AW1IPState1
```

29 行目からエージェントラッパー AW1 が実行するプログラムを記述している。31 行目から 40 行目までが仲介エージェント”Bee1”を作成するための記述である。36 行目以降はもしも仲介エージェントを作成する段階でエラーが起きた場合、その旨を表示してプログラムの実行を停止するためのものである。
42 行目からは仲介エージェントからの要求を待ち、要求が来たときにどのような対処を行うかが記述されている。XML/ACL メッセージ（Bee-gent で用いられるエージェント間の会話言語）のタグの内容を取得するために、メソッド get2Tag2Value() を使用する。この例では 44 行目でタグ”performative”の値を取得している。46 行目から 48 行目は”performative”の値が”request”であった場合、同様に”action”、”args”の値を取得している。そして、”action”の値が”say”であるなら、”args”に与えられた情報を表示する、というものである。

● HelloWorld プログラムの仲介エージェント例

```
 1  import java.io.*;
 2  import java.net.*;
 3  import java.util.*;
 4
 5  import com.toshiba.beegent.*;
 6  import com.toshiba.beegent.util.*;
 7  import com.toshiba.beegent.xml.*;
 8
 9  public class Bee1 extends Bee implements I_Bee{
10
11    public void init() throws InstantiationException{
12
13      addIPStates(new BeeIPStateINIT());
14
15      //addPublicIPStates();
16
17    }// init
18
19  }// Bee1
20
21  class BeeIPStateINIT extends BeeIPState implements I_BeeIPState{
22
23    BeeIPStateINIT(){
24      setPrecond("INIT");
25    }// constructor
26
```

エージェントラッパーと同様に、1 行目から 19 行目までが仲介エージェントのクラス定義である。9 行目で仲介エージェントの名前を Bee1 としてクラスを定義している。また、13 行目で登録した BeeIPStateINIT のクラスを 21 行目から定義している。24 行目で事前条件を”INIT”に定義している。事後条件が定義されていないが、これは後で定義される。

```
27    public void action(){
28
29      // Send XML/ACL
30      XmlAcl xs = new XmlAcl();
31      xs.setTag2Value("performative", "request");
32      xs.setTag2Value("sender"      , "Bee1");
33      xs.setTag2Value("receiver"    , "AW1");
34      xs.setTag2Value("action"      , "say");
35      xs.setTag2Value("actor"       , "AW1");
36      xs.setTag2Value("args"        , "Hello World!");
37      xs.setTag2Value("protocol"    , "");
38      xs.setTag2Value("reply-with"  , "");
39      xs.setTag2Value("ontology"    , "");
40      if(!sendXML(xs)){
41        setPostcond("END");
42        return;
43      }
44
45      // Post
46      setPostcond("END");
47
48    }// action
49
50  }// BeeIPStateINIT
```

27 行目から実際に仲介エージェントがどのような行動をするのかが記述されている。30 行目から 39 行目で XML/ACL メッセージのタグを設定する。タグの種類は”performative”の種類ごとに違うが、

- sender(送信者)
- receiver（受信者）

上記の二つのタグは共通である。

前述のエージェントラッパー AW1 の行動は”performative”が”request”であり、”action”の内容が”say”であった場合、”args”の内容を表示する、というものであった。この例で言えば、仲介エージェントの 36 行目の”args”を書き換えれば任意に表示する文章を変えられる。

3.2 Bee-gent による仲介システム

著作権の詳細を決定する権利はあくまで人間である製作者にあるため、プログラムの自己判断だけで変更して良いものではない。そのため、Bee-gent に仲介という役目を持たせる。具体的には利用者側のエージェントラッパー、製作者側のエージェントラッパー、その間の仲介エージェントを作成し、それぞれに役目を持たせる。

利用者側のエージェントラッパーにはコンテンツの二次利用を製作者に要求する旨を伝えるための役割を持たせる仲介エージェントを作成し、その後製作者側の返事を待つようにし、その返事（許可・不許可）を表示する。製作者側のエージェントラッパーにはそれに対しての返事を用意し、仲介エージェントを介してそれを利用者に伝える役割を持たせる。

3.2.1 利用者側のエージェントラッパー

```
publpublic void action(){
  // Receive XML/ACL
  while (waitXML(0)){
   XmlAcl xa = getXML();
    String perf = xa.getTag2Value("performative");
   if (perf.equals("agree")){
    String action = xa.getTag2Value("action");
    String args   = xa.getTag2Value("args");
   if (action.equals("permission")){
     System.out.println("The request was permitted. ");
     System.out.println(args);
        }
   if (action.equals("disapproval")){
     System.out.println("The request was refused.");
     System.out.println(args);
   }
```

この記述により、仲介エージェントから受け取る ACL によって二次利用が許可されたか否かを表示する。また、製作者側が”args”に文章を記述することで、製作者のコメントとして表示することも可能。

3.2.2 利用者→製作者の時の仲介エージェント

```
  publpublic void action(){
    // Send XML/ACL
    XmlAcl xs = new XmlAcl();
    xs.setTag2Value("performative", "request");
    xs.setTag2Value("sender"      , "User");
    xs.setTag2Value("receiver"    , "Creator");
    xs.setTag2Value("action"      , "reply");
    xs.setTag2Value("actor"       , "Creator");
    xs.setTag2Value("args"        ,
     "Would you accept secondary use of these contents?");
    xs.setTag2Value("protocol"    , "");
    xs.setTag2Value("reply-with"  , "");
    xs.setTag2Value("ontology"    , "");
    if(!sendXML(xs)){
      setPostcond("END");
      return;}
```

利用者の要求を製作者側エージェントラッパーに伝えるための記述。”args”の部分にどの作品をどのように二次利用するのかを記述すればその意思を製作者に伝えられる。

3.2.3 製作者側エージェントラッパー

```
 public void action(){
    // Receive XML/ACL
    while (waitXML(0)){
      XmlAcl xa = getXML();
      String perf = xa.getTag2Value("performative");
      if (perf.equals("request")){
String action = xa.getTag2Value("action");
String args   = xa.getTag2Value("args");
if (action.equals("reply")){
  System.out.println(“ There was a request from a user. ");
  System.out.println(args);
        }
```

利用者からの要求を待ち、要求があった場合それを表示させるための記述。利用者側のエージェントラッパーと同様、”args”にコメントが書かれていればそれを参照できる。

3.2.4 製作者→利用者の時の仲介エージェント

```
publpublic void action(){
  // Send XML/ACL
  XmlAcl xs = new XmlAcl();
  xs.setTag2Value("performative", "agree");
  xs.setTag2Value("sender"      , "Creator");
  xs.setTag2Value("receiver"    , "User");
  xs.setTag2Value("action"      , "permission");
  xs.setTag2Value("actor"       , "User");
  xs.setTag2Value("args"        , "Please use freely. ");
  xs.setTag2Value("protocol"    , "");
  xs.setTag2Value("reply-with"  , "");
  xs.setTag2Value("ontology"    , "");
  if(!sendXML(xs)){
    setPostcond("END");
    return;}
```

製作者側の返事を利用者に伝えるための記述。利用者→製作者の時の仲介エージェントと同様、”args”にコメントを記述すればより詳細な返事が可能となる。

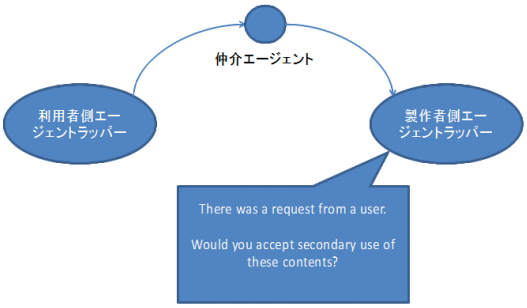

図 3.1 利用者→製作者の時の動作

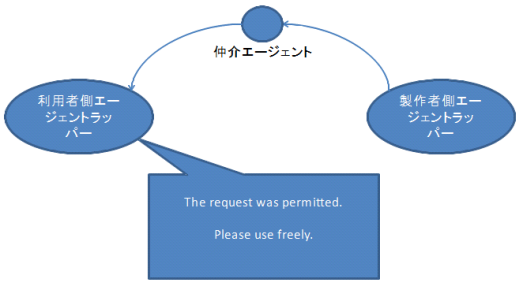

図 3.2 製作者→利用者の時の動作

第4章

結論

本稿では、Bee-gent による仲介システムとその一例を提案した。序論で説明したがインターネット時代の現在において、著作物と、それを製作する著者の数は膨大であり、それらの権利を一つのシステムが管理するのは難しい。また、今まで提供されるものを享受するだけの立場だった公衆が製作者側にまわる事も多くあるため、「自身の作品の権利を自身で管理する」必要性が出てきてしまっている。そこで、本研究では権利管理が大変になってしまったこのインターネット時代という環境を逆に利用し、利用者は気軽に製作者にコンタクトを取ることが出来、製作者もその返事を容易に行えるシステムを提案するに至った。

今後は Bee-gent の XML/ACL に新しく著作権を扱うための言語を設定し、仲介だけに留まらない著作権管理のためのエージェントを作成する。

謝辞

本研究を行うにあたり、終始熱心にご指導して頂いた木下宏揚教授に深く感謝いたします。さらに、公私にわたり良き研究生活送らせて頂いた木下研究室の方々に感謝いたします。

2012 年 2 月
山口　誉央

参考文献

[1] ”インターネット時代の著作権問題”
http://www.law.co.jp/okamura/copylaw/A.HTM

[2] ”インターネット時代の著作権　小倉秀夫”
http://www.rieti.go.jp/jp/events/03042101/pdf/ogura.pdf

[3] ”著作権の歴史”
http://cozylaw.com/copy/tyosakuken/history.htm

[4] ”IT 用語辞典”
http://e-words.jp/

[5] ”対談：デジタル著作権ってどうなってるの？”
http://japan.cnet.com/news/commentary/20384798/

[6] ”Microsoft Windows Media　デジタル著作権管理（DRM)”
http://www.microsoft.com/japan/windows/windowsmedia/drm/default.aspx

[7] ”ソフトウェアエージェント特集”
http://img.jp.fujitsu.com/downloads/jp/jmag/vol50-4/paper05.pdf

[8] ”クリエイティブ・コモンズ”
http://creativecommons.jp/

[9] ”クリエイティブ・コモンズのメタデータ　著作権とライセンス記述の新しい形”
http://www.kanzaki.com/docs/sw/ccm.html

[10] ”Digital Right Description としての CC ライセンス”
http://www.baldanders.info/spiegel/log2/000349.shtml

[11] ”コンテンツビジネスのテクノロジー　DRM の有効性”
http://i.impressrd.jp/files/images/bn/pdf/im200303-154-DRM.pdf

[12] ”自由で安全なコンテンツ流通を実現するエージェントフレームワーク Freedia の公開”
http://www.nii.ac.jp/kouhou/smartive20070608.pdf

[13] ”スマーティブプロジェクト”
http://www.saitama-j.or.jp/sangaku/seeds/dk/ooita/OU006.pdf

[14] ”たのしい XML”
http://www6.airnet.ne.jp/manyo/xml/

[15] ”RDF/XML 構文の簡単な説明”
http://www.kanzaki.com/docs/sw/rdf-xml.html

[16] ”Bee-gent マルチエージェントフレームワーク”
http://www.toshiba.co.jp/rdc/beegent/ 最終閲覧日 2012 年 2 月 8 日

質疑応答

- 提案したような記述をユーザーも製作者も記述しなければならないのか？（松澤先生）

重要なのはXML/ACLのタグ（performative、Senderなど）の値なので、その値を変えればメッセージの意図も、送り主も送り先も、動作内容やコメントも変更する事が出来るので、それらを入力するインターフェースが作成出来れば、記述は容易になる。

- Bee-gentは日本語には対応していないのか？（豊島先生）

デフォルトでは対応していない。（タグの値に日本語を入力してもエラーが出る）各言語に対応させるのも今後の課題とする。