DDSS　UI 改善プロジェクト

最終ドキュメント

PM
竹元和彦さん（日本 IBM）

メンバー
瀬端博志（環境情報学部４年）
川口将司（環境情報学部２年）
安藤亮一（環境情報学部３年）
荒木 恵（総合政策学部４年）

DDSS 最終ドキュメント：目次

1. 企画

この章では DMC ユーザーインターフェース改善プロジェクトの企画について述べる。

1.1. 企画概要

本プロジェクトは、DMC(慶応義塾大学デジタルメディア・コンテンツ統合研究機構)による、セマンティック検索システムのユーザインターフェースを改善するプロジェクトである。本プロジェクトの構成メンバーは以下の通りである。

PM:竹元和彦さん（日本 IBM）
メンバー：瀬端博志（環境情報学部４年）
川口将司（環境情報学部２年）
安藤亮一（環境情報学部３年）
荒木 恵（総合政策学部４年）

1.2. ヒアリング

本プロジェクトの顧客である DMC から、嶋津恵子先生にお越しいただき、ヒアリングを行なった。

1.2.1. 第一回ヒアリング（2006/04/27）

第一回のヒアリングでは、質問項目を決めてのヒアリングというよりは、顔合わせのような意味合いで、話をしながらクライアントが何を求めているのかを聞いた。
この日にクライアントから得られた情報は以下の通りである。

1.2.1.1. システムについて

・このシステムはコンテキストとしてドキュメントを連結させて検索するシステムである。
・探したいドキュメントが決まっていない、キーワード群が分からない人向けのシステムである。
・ただの検索ではなく、「つながり」を検索できるシステムである。

1.2.1.2. メタデータ抽出方法について

・RDF などで埋め込まれたメタデータ
・文書から自動抽出されたメタデータ
の２種類がある。

1.2.1.3. DDSS 班に対しての要求

・クライテリアの設定
・「つながり」を検索できるシステムであることが分かる UI
・できれば、英語版の UI の作成

1.2.1.4. 質疑応答

Q　UI のどのあたりに問題があると考えているか？
A　通常の検索システムに見えてしまう。
次にどんな操作をすれば特徴的な使い方ができるかが直感的にわからない。
画面遷移が複雑。

Q　特にどのページに問題があると考えているか？
A　やはりトップページか。
画面数や遷移方法なども変えてもよい。

Q　どうやってコンテンツを探しているのか？
A　シーズ（リスト）にはいっているサーバから 16 階層までを検索。（ほぼ全てを網羅）
WEB への UP ができない人には、登録できるシステムを用意。
可能であればプロジェクトメンバーもその登録システムを利用できるように。

Q　スコアで表示される値の詳細は見れるのか？
A　合計値としてしかわからないため、難しい。

Q　メタデータの埋め込み方法や、どのメタデータを埋め込むべきかなどの情報は？
A　現状ではそういった説明のページはない。
ぜひ作ってほしい。

Q　発想法でいうなら、大量のカードがある状態？
A　カードの並べ方（「島」づくり）を支援するシステム。
発想法という言葉を使ってしまうと、議論が起こってしまう。
そこに巻き込まれるのは望ましくない。
同じような定義だが、新しい概念としてうまく受け入れられるように「見せられ」ないだろうか？

1.2.2. 第二回ヒアリング(2006/05/11)

第一回のヒアリングを受けて、私たちは、メタデータの種類による色分け表示や、チュートリアルの作成、検索ボタンやメタデータ抽出ボタンを見えやすい位置に置く、無駄なテキストボックスを消す、などの方法を考えて、第二回のヒアリングに臨んだ。
しかし、第二回のヒアリングにおいて、クライアントの要求は、「現在の UI の改善」というよりは、「全く新しい UI の提案」であることが分かり、私たちは、DMC の UI に対する

考え方の転換をはかることになった。
第二回のヒアリングで得られた情報は、以下の通りである。

1.2.2.1. クライアントの要求

エンドユーザーが使いやすくて活用できる
こんな風に使える、活用できると示す
学生による、開発側が思いもよらない使い方
アイデアの取捨選択は担当できる
コンテンツに対する色々な軸があるということが示せるように
もっと自由なＵＩ変更なども思いついてほしい
新しい発想が欲しい
こちらは単に使いやすくを重視していたが、魅力的にみせる工夫も大事

1.2.2.2. チュートリアルの作成について。

・チュートリアルは必要ないのではないか。
　google や yahoo の使い方を先に読む人はいないという統計があり、
　読むのは使った後に活用しようと考えた人がほとんど
　チュートリアルがそれでも必要というならデータで示して欲しい

1.2.2.3. このプロジェクトの進め方に対して

・用語定義は長い時間かけても仕方が無い。意思統一のためならばもっと短い時間で。
・全体が決まっていない。全体を決めて欲しい。
・ゴールを決めるべきである。
・部分的で全体が見えない。木を見るか森を見るか。

1.2.2.4. ヒアリングの日に出されたアイデア

これらのヒアリング結果を受けて、ヒアリング当日にクライアントである嶋津先生と私たちは、以下のようなアイデアを出した。
・登録コンテンツ数が少ない⇒メタデータ登録ツールの作成（メタデータの登録を自動化）
・ただの検索エンジンに見えてしまう
　⇒GoogleMap のような形式での地名メタデータの表示
　⇒年表などによる時のデータの表示
　⇒人名マップの作成（GoogleMap のようにできないか？）
　⇒ベン図表示で見やすく。
・使い方が分かりにくい
　⇒読んでもらえるウィザード風チュートリアルの作成

1.2.2.5. メタデータについて

また、前回はっきりと定義できなかった「メタデータ」について、島津先生の意見を伺った。

・メタデータは記憶のトリガである。
・共通項目で引けるメタデータ（例：同時代に生きた人と、その当時の文化、食べ物など）
・自動登録されているメタデータの種類（人・場所・時間）
・メタデータは５W1H＋αに分類されている。

1.2.2.6. その他

その他、以下のような議論が成された。

・ユーザ観点から見て、「メタデータ」という語はどのような意味を持つか？
　（切り口？具体的にメタデータに入る情報はなに？など）
・「人」というメタデータと「時」というメタデータはどちらが大切か
　⇒どこを売りにするかによって変わる余地はある。
・ツリー構造を構築して好きに変更できるうようなイメージ？
　⇒実装するとなると量に耐え切れない。
・どうして使いにくいと感じるのか？
　⇒原因究明の必要性

1.2.3. 第三回ヒアリング(2006/05/24)

第二回のヒアリングを受けて、私たちは、このプロジェクトに求められているものが検索システム UI の小さな変更ではなく、学生の視点からまったく新しい UI を提案することだと理解した。私たちはこのシステムが使いにくい、分かりにくいと感じる原因を分析し、議論を行い、「ベン図検索システム」というアイデアに行き着いた。第三回ヒアリングは、ベン図検索システムの概要を説明し、それに対しての意見を伺うという形で行なった。

1.2.3.1. ベン図検索システムについて

・既存の（東大の）ベン図システムと比較して、ベン図検索システムのアドバンテージとは？
　⇒自分で検索ワードを考える・見つける必要はない。システム側から自動的に提供してくれる点がアドバンテージ。
・ベン図の数の問題。
　⇒実装の技術的にもユーザーインターフェース的にも高々3 つが限度。
　　かなり実装が難しそうである。
・概念旅行というフレーズ
　⇒良い。Google の様な検索システムはユーザーに対し、『勝手に旅行して下さい』的な態

度。これに対してベン図検索システムはベン図というものを使って、旅行をサポートしている。
このように、ベン図検索システムのアイデアについては、おおむね好評であった。

1.2.3.2. プロジェクトの進め方について

・早い時期に実装しないとテスト期間が設けられない。単体・結合テストも頭に入れて。
・クライアントが 1 年間かけても良いと言ったとしても、研究会だから短期間で仕上げないといけない。
・取り掛かる前に、まずフィージビリティを持たせなくてはならない。一週間の間に誰が何時間動けるか、技術的なものはどうか等。今のところ、このフィージビリティがない。
・状態遷移・画面遷移図を早めに作成する。
・開発の設計は竹元さんに頼った方が良い
・システム開発においてプロトタイプというのはありえない。良い部分はそのまま継承すべき。（ウォーターフォール）ダメなところだけ作り直すこと。
・全体を反復するということはあまりしない。

1.3. 提案書作成

これらの３回のヒアリングを受けて、私たちは提案書の作成に取り掛かった。提案書の作成の間に、

・要件がはっきりしていない
・ゴールの設定ができていない

などのレビューを受け、議論の結果、最終的に提案書を提出したのが 2006/07/13 であった。
提出した提案書は以下の通りである。

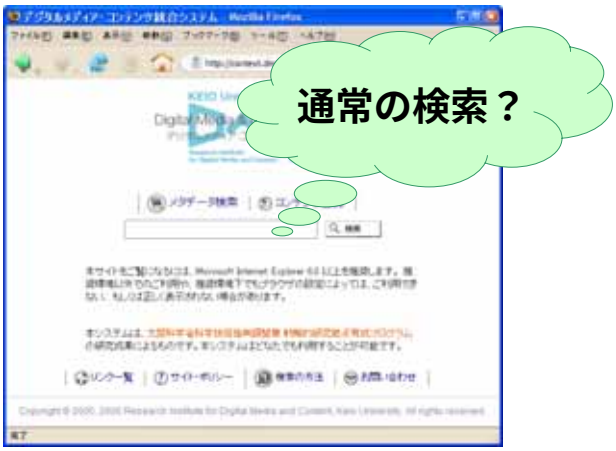
目次

- 要件
- 現状の分析
- 提案

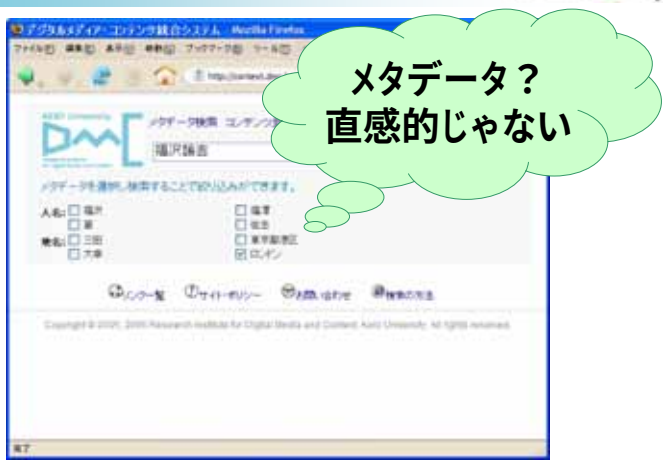
要件

- 既存の「デジタルメディア・コンテンツ統合システム」のユーザー・インターフェースの改善
  - “発見”・“発想”を支援するためのシステムだということがわかるようなUI
  - パソコンに不慣れなユーザーでも直感的に使えるようなUI
  - 魅力的で新しいUI
- ⇒**『DMCに初めて触れる文学部の学生がヘルプを見ることなくセマンティック検索が出来るようにするUI』**

現状の考察

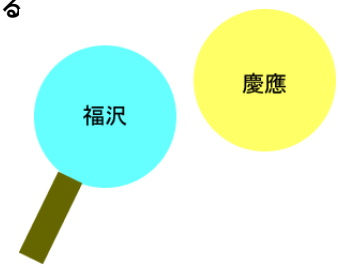


現状の考察



提案

- 虫眼鏡でのぞきこむ検索
  - 検索語を集合の円で、メタデータを虫眼鏡で表現する
  - 「検索語“慶應”を含み慶応に関するメタデータ“福沢”をもつコンテンツを検索する」といった状況を「虫眼鏡“福沢”で検索語の集合“慶應”を覗き込む」という形で表現する



このUIの特徴

- 「関係がある」ということを「円の重なり」という形で表現し、理解を促している点
- 語句を組み合わせる検索を視覚的に行うことで、感覚的に利用できるようにしている点
- 「虫眼鏡で検索語を覗き込む」というメタファーを用いることで、理解しやすくしている点

トップページ

□検索語入力バー

検索語を入力する領域

@「検索語入力バー」に検索語を入力し、「検索ボタン」をクリックする

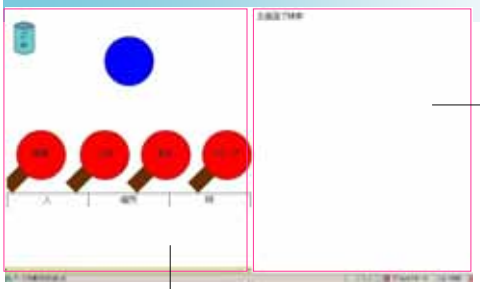
入力された検索語で検索をする

⇒検索初期画面へ遷移

□その他メニュー

サイト説明や簡易マニュアル等へのリンク、またその他

検索初期画面



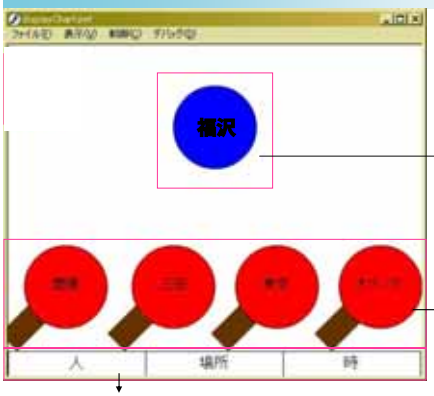
□検索結果表示領域

Flash上で検索のアクションが起こった時に、検索結果が表示される領域

□虫眼鏡で覗き込む検索モジュール

検索を行うためのFlashモジュール

Flash検索画面(初期画面)



□検索語集合

入力された検索語の集合を現す円

□関連語集合

検索語に関連している語句の集合が表示される

現状のDMCのシステムで関連語は各項目において4つと制限されているため、それに従い、各項目で4つまで表示する

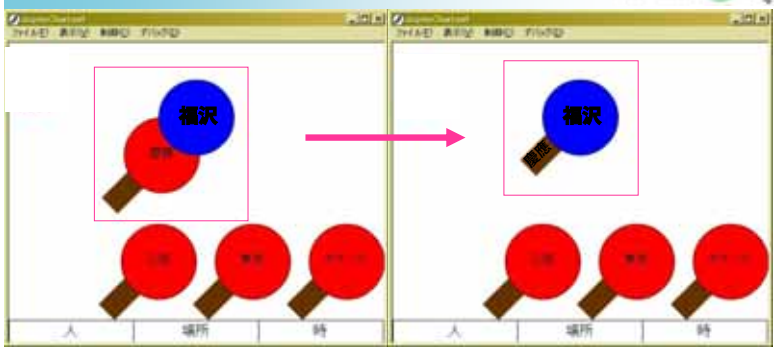
□関連語項目切り替えボタン

表示される関連語の項目を切り替えるためのボタン

@「関連語項目切り替えボタン」をクリックする

・表示される関連語の項目が押したボタンの項目に切り替わる

Flash検索画面(メタデータを重ねる)



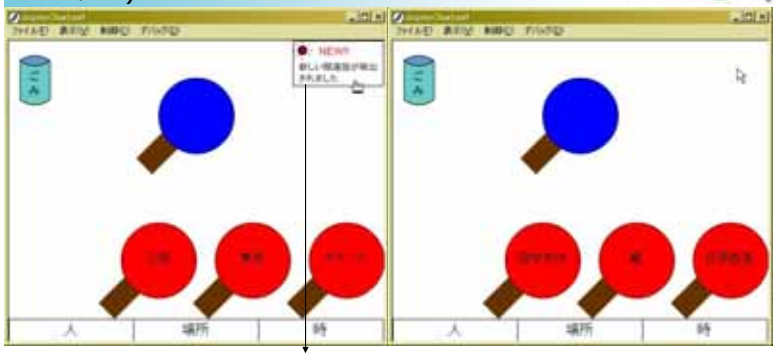
@「関連語集合」を「検索語集合」の上にドラッグ＆ドロップする

・関連語を検索語と重ね合わせる

・検索語と重ねられた関連語をキーワードにセマンティック検索を行う

⇒検索結果表示領域に検索結果を表示

Flash検索画面(新しいメタデータを抽出する)



□関連語抽出ボタン

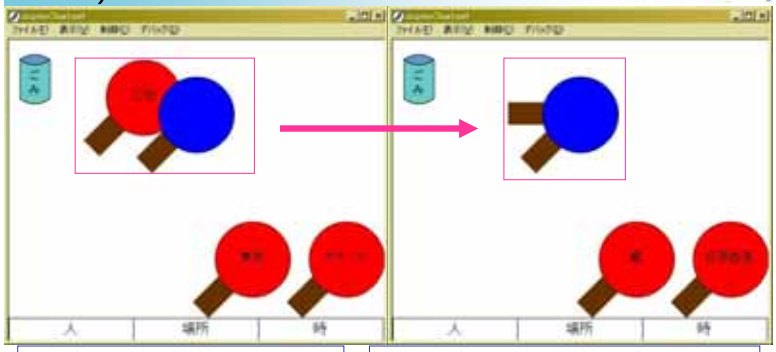
検索結果から新たな関連語が発見された場合、表示される

新たな関連語を抽出するボタン

@「関連語抽出ボタン」をクリックする

・現在表示されている関連語集合を、新たに発見された関連語に更新する

Flash検索画面(さらにメタデータを重ねる)



@既に「関連語集合」が重なっている「検索語集合」の上に別の「関連語」をドラッグ＆ドロップする

・検索語と重なっている全ての検索語をキーワードにセマンティック検索を行う

⇒検索結果表示領域に検索結果を表示

@重なっている「関連語集合」を抜き取る（「関連語集合」の枠を持って「検索語集合」の外でドラッグ＆ドロップする）

・検索語と抜き取った関連語を除いた重なっている全ての関連語をキーワードにセマンティック検索を行う

⇒検索結果表示領域に検索結果を表示

新UIに取り入れるDMCの機能

取り入れる部分

キーワード検索

関連語抽出

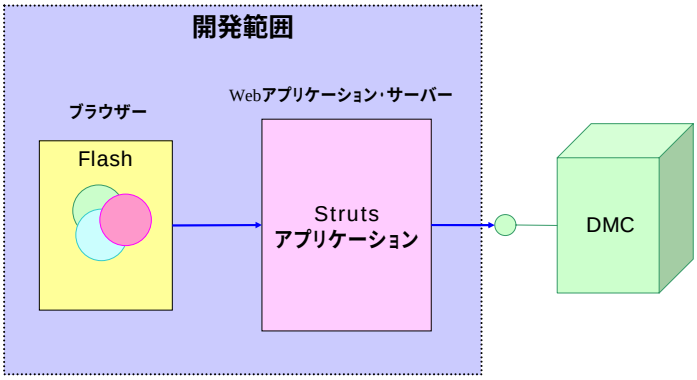
キーワード
＋抽出関連語検索
(セマンティック検索)

取り入れない部分

関連語検索
(メタデータ検索)

コンテンツ登録
(非検索機能)

開発範囲



評価

- 評価者
  - 大岩研究会Ⅰの学生
  - DMCの方々
  - 文学部の学生
- 評価方法
  - ユーザーテスト
    - テストの様子の観察・記録
  - アンケート
  - 感想を聞く

評価

- ユーザーテスト
  - 大岩研究会Ｉの学生によるユーザーテスト（7月24日）
  - DMCの方々によるユーザーテスト（予定調整を行なう）
- 評価基準
  - チェックリストが基準点（後述の評価基準で、0点が10人に1人以下、平均点が2．5点以上）を満たす

評価基準

- 0…Flashが表示された後の使用方法がわからない（虫眼鏡をドラッグして検索語に重ねられない）
  - ⇒要件を満たしていない状態
- 1…虫眼鏡をドラッグして検索語にかさねられ、検索結果のHTMLをクリックできた
  - ⇒要件を満たした状態
- 2…メタデータ属性変更ボタンを押すことができた
  - ⇒広い意味での要件を満たした状態
- 3…1の操作を2回以上繰り返した
  - ⇒興味を持てるUIかどうか
- 4…インストラクションで指定された検索語以外の言葉でセマンティック検索をしてみた
  - ⇒自分なりの使い方（楽しみ方）ができるUIかどうか
- 5…セマンティック検索の結果の自分なりの使用法を思いつくことができた。
- 6…メタデータの意味（関連がある言葉が出てきている）がわかった。

このような流れで、企画フェーズを終了した。

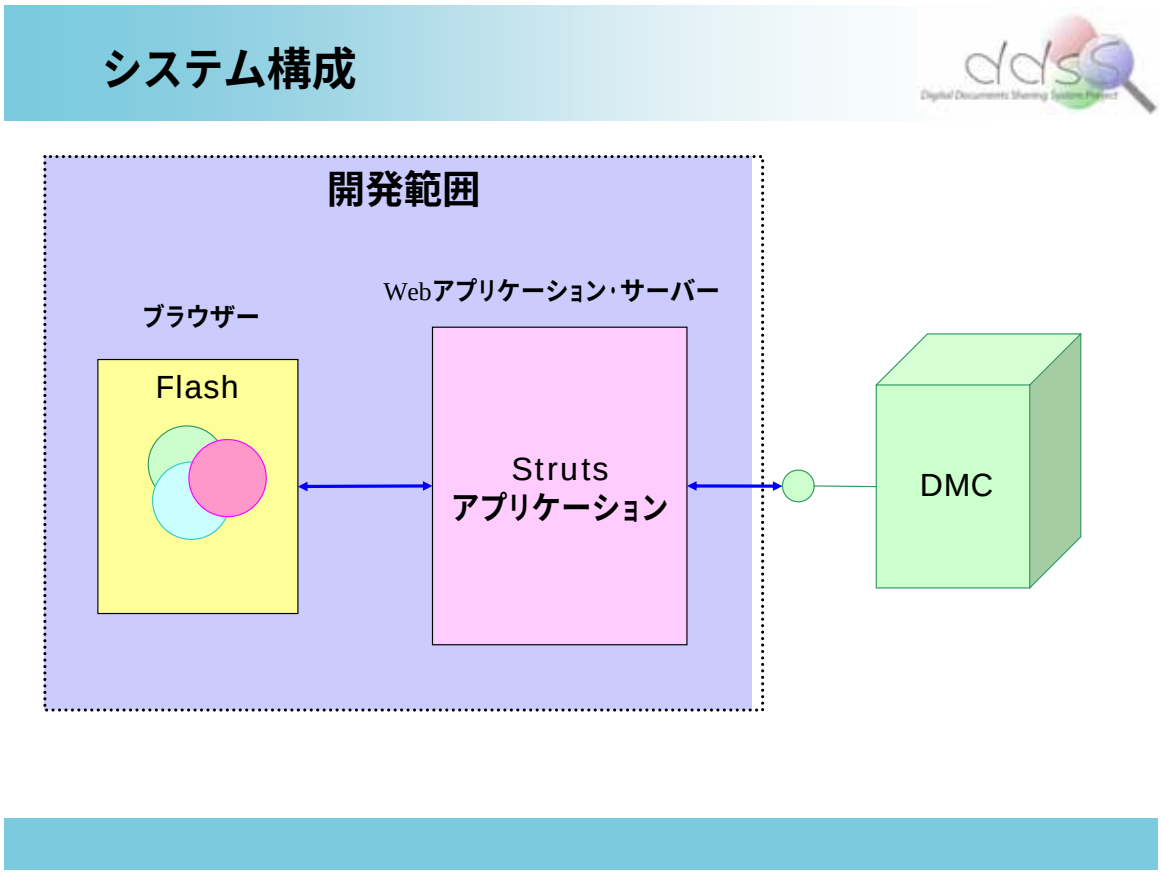
初期の予定では第一回の反復の時点で企画フェーズは終了する予定であったが、要件定義やスコープの決定に時間がかかったため、企画フェーズが実装期間とかぶる形になってしまった。実際には、企画の洗練と、設計・実装は重なった時期（６月後半～７月前半）に行なわれた。

2. 設計

2.1. システム構成図

今回のプロジェクトで作成するシステムの全体構成を明らかにし、スコープを明確にするため作成した。

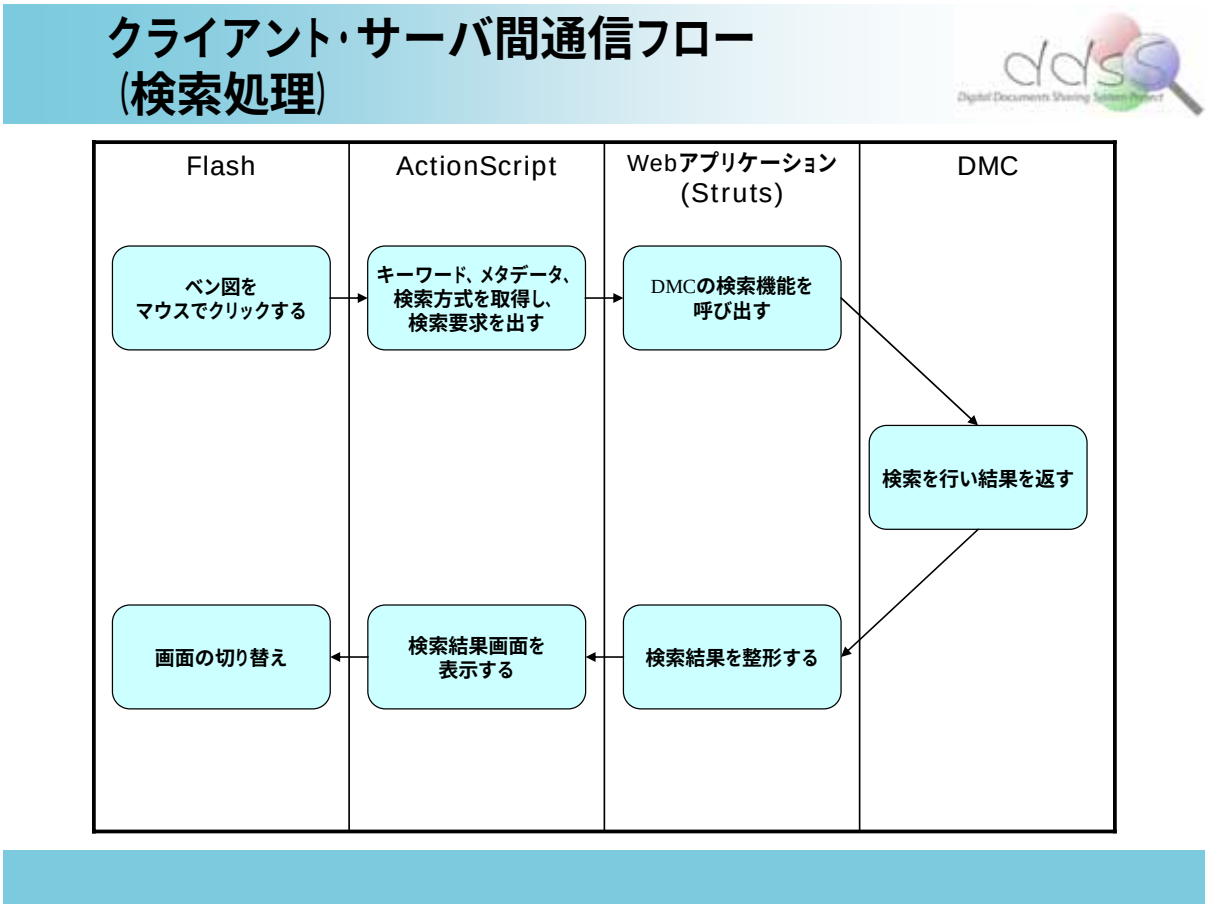
⇒図１　システム構成図



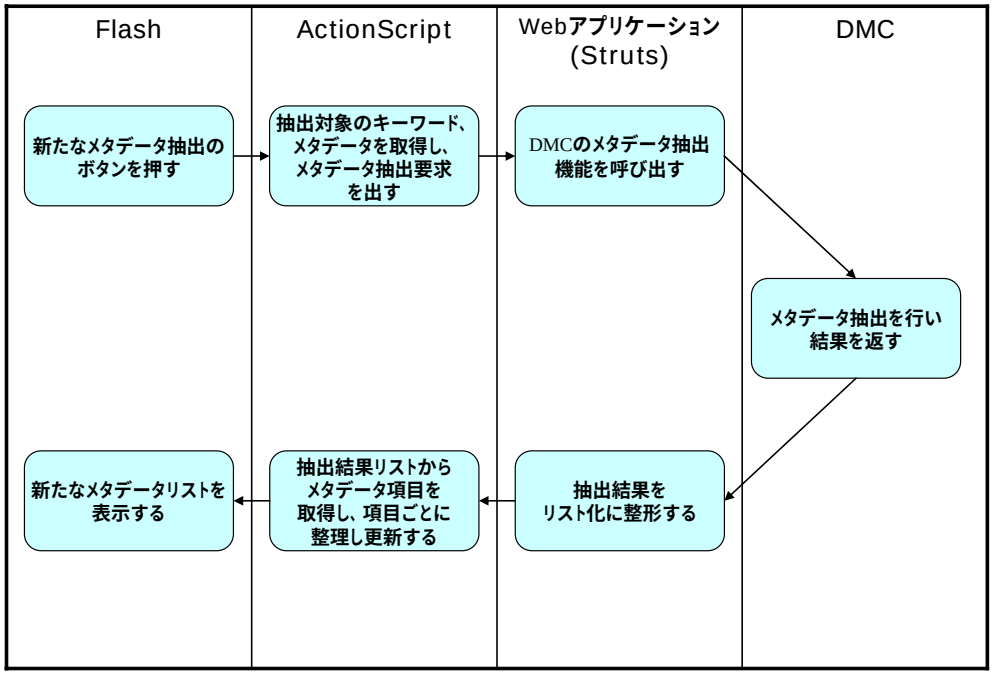
2.2. クライアント・サーバ間通信フロー

クライアントとなる Flash とサーバとなる Struts 間でのデータ通信の流れと、通信処理の数を把握するために作成した。

⇒図 2　クライアント・サーバ間通信フロー



クライアント・サーバ間通信フロー
(メタデータ取得処理)



2.3. 通信プロトコル規定

クライアントとなる Flash とサーバとなる Struts で通信を行う際、どのような形式でデータをやり取りするかを明確にするため作成した。

[通信プロトコル規定]

Flash からの要求と Struts からの応答はどちらも HTTP 上で実現

1.検索処理

・Flash からの要求

・パラメータ

・command=”search”

・queryString=”検索語”

・checkedList=”メタデータ”

⇒多数のメタデータがある場合、カンマ区切りで送信

・Struts からの応答

・HTML

・結果ページのタイトル

・結果ページの URL

・結果ページのサマリー

2.メタデータ取得処理

・Flash からの要求

・パラメータ

・command=”relation”

・queryString=”検索語”

・checkedList=”メタデータ”

⇒多数のメタデータがある場合、カンマ区切りで送信

・Struts からの応答

・XML

・メタデータ名

・メタデータ種類

[XML スキーマ]

```
<?xml version="1.0" encoding="utf-8" ?>
- <metadata>
<metaword type="Person">大学</metaword>
<metaword type="Person">黒田和也</metaword>
```

```
<metaword type="Person">酒井 政裕</metaword>
<metaword type="Person">眞田</metaword>
<metaword type="Place">ソフィア</metaword>
<metaword type="Place">川崎</metaword>
<metaword type="Place">三田</metaword>
<metaword type="Place">東京</metaword>
</metadata>
```

2.4. Actionscript クラス図

Flash 内の Actionscript の静的構造を明確にし、クラス構造を明らかにすることで実装を進めやすくするために、クラス図を作成した。

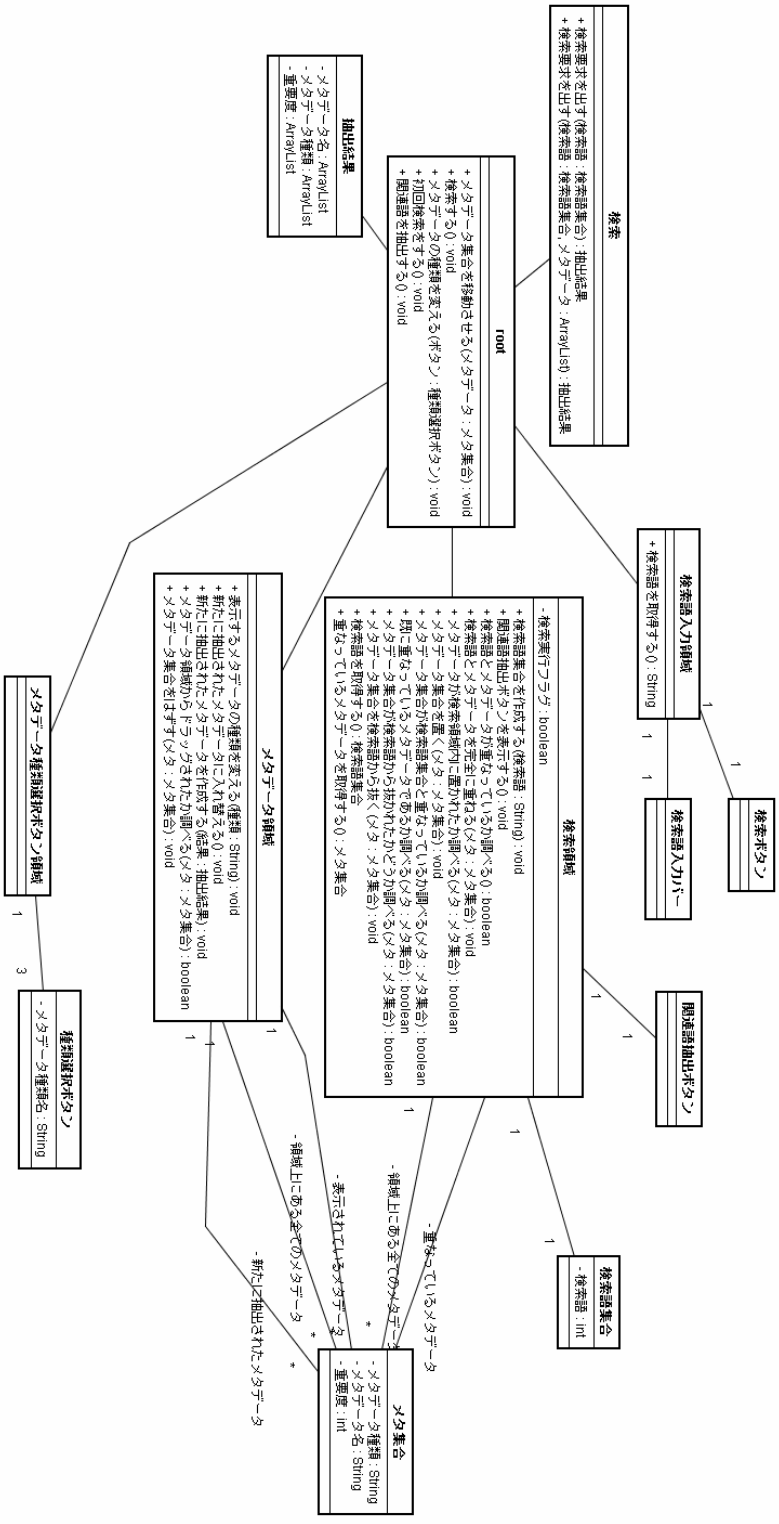


図１：ActionScript クラス図

2.5. Actionscript シーケンス図

Flash 内の Actionscript の動的振る舞いを明確にし、オブジェクトが持つ操作の呼び出し手順を明らかにすることで実装を進めやすくするために、シーケンス図を作成した。

⇒図 4　Actionscript シーケンス図

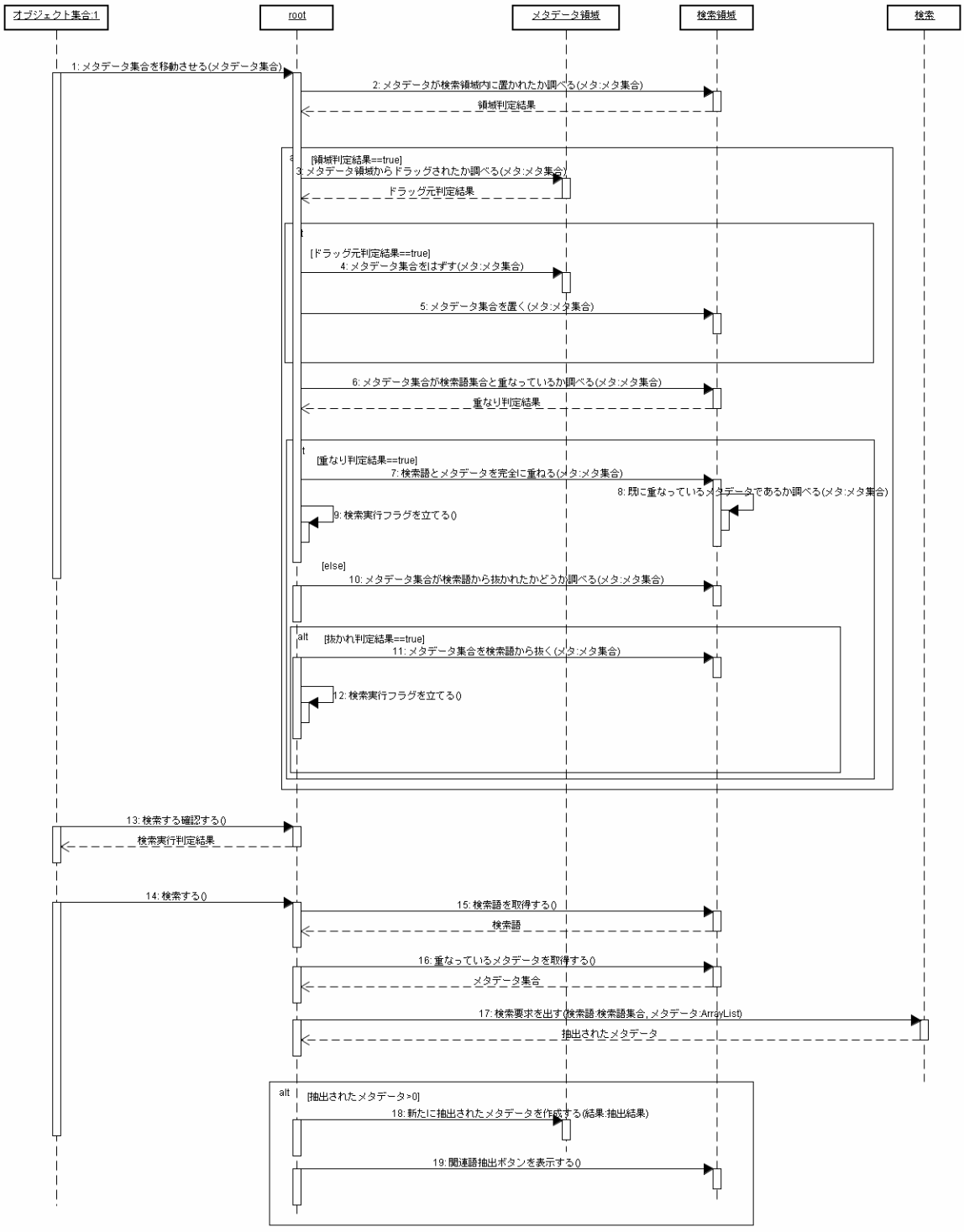


図２：メタデータをドロップ時の振る舞い

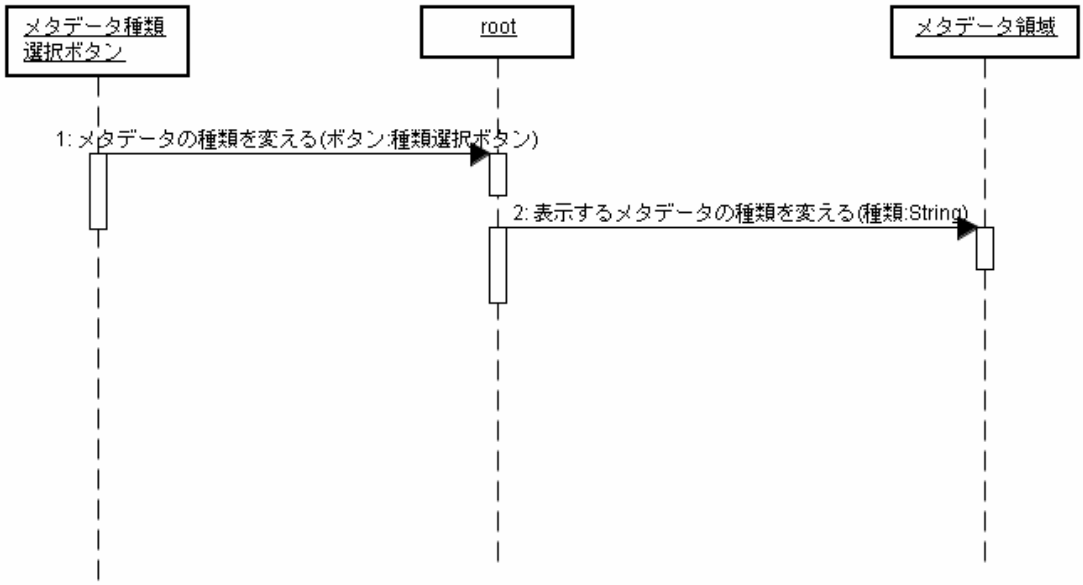


図３：メタデータ種類選択ボタンをクリックした時の振る舞い

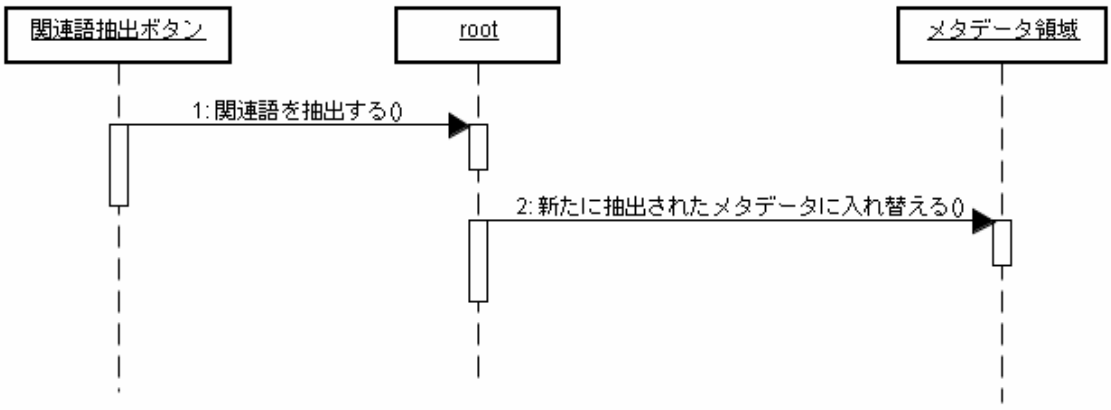


図４：関連語抽出ボタンをクリック時の振る舞い

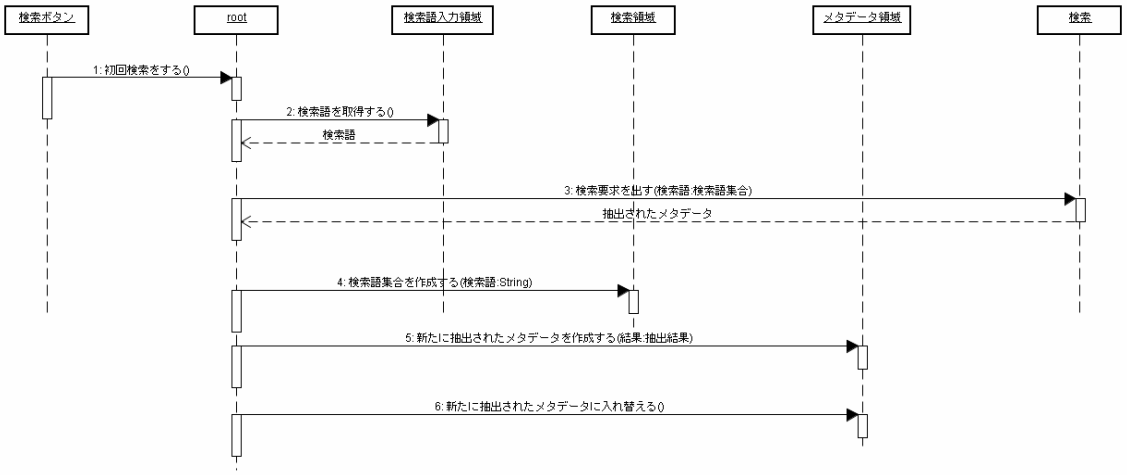


図５：検索ボタンをクリック時の振る舞い

3. 実装

3.1. クライアントサイド実装

3.1.1. 開発環境

Flash Professional 8

3.1.2. 実装項目

Flash アプリケーション
ハイパーテキスト

3.1.3. ソースコード

●main.fla（メインクラス）

```
import mx.utils.Delegate;



/*クライアント*/
var order:Number = 1;
var searchTerritory:SearchTerritory;
var newSearch:Boolean;

//検索ボックス
var textbox:TextField = _root.createTextField("inputBox", 9999, 100, 5, 230, 20);
var textFormat:TextFormat = new TextFormat();
{
	textFormat.font = "Arial";
	textFormat.color = 0x000000;
	textFormat.size = 14;
}
textbox.setNewTextFormat(textFormat);
textbox.selectable = true;
textbox.border = true;
textbox.type = "input";

//検索ボタン・・・(仮）物
var searchButton:MovieClip = _root.attachMovie("search_button_mc", "search_button", 10000);
```

```
searchButton._x = 340;
searchButton._y = 7;
searchButton.onRelease = Delegate.create(this, released);
function released() {
	if(textbox.text != "") {
		newSearch = true;
		getSpell(textbox.text);
	}

	searchTerritory.deleteMetaInSearchTerritory();
	searchTerritory.setVisible(false);
}

//属性変更ボタン
var elementButton:Array = new Array(3);
var elementTextFormat = new TextFormat();
var elementText:TextField = new TextField();
var elementTextList:Array = new Array(3);

{
	elementTextFormat.bold = true;
	elementTextFormat.size = 14;
	elementTextFormat.font = "Arial";
	elementTextFormat.color = 0xFFFFFF;
	elementTextFormat.align = "center";
}

var ELEMENT_BUTTON_Y:Number = 469;

for(var j:Number = 0; j < elementButton.length; j++) {
	elementButton[j] = _root.attachMovie("element_mc", "elementButton", 5500 + j);
	elementButton[j]._x = elementButton[j]._width * j;
	elementButton[j]._y = ELEMENT_BUTTON_Y;
	elementTextList[j] = this.elementButton[j].createTextField("elementText" + j, 5503 + j, 0, 5, 167, 3
	elementTextList[j].setNewTextFormat(elementTextFormat);
	switch(j) {
```

```
                    case 0:
                    elementTextList[j].text = "人";
                    break;

                    case 1:
                    elementTextList[j].text = "場所";
                    break;

                    case 2:
                    elementTextList[j].text = "時";
                    break;
            }
            elementTextList[j].selectable = false;

            //elementButton[j].onRelease = elementAction(elementTextList[j].text);
}

elementButton[0].onRelease = function() {
            metaTerritory.changeMetaKind("Person");
}
elementButton[1].onRelease = function() {
            metaTerritory.changeMetaKind("Place");
}
elementButton[2].onRelease = function() {
            metaTerritory.changeMetaKind("Time");
}

//検索領域内にドロップされたかを調べる
function dropCheckX(metaSet:MetaSet, metax:Number):Number {
            metax = searchTerritory.dropCheckX(metaSet, metax);

            return metax;
}

function dropCheckY(metaSet:MetaSet, metay:Number):Number {
            metay = searchTerritory.dropCheckY(metaSet, metay);
```

```
		return metay;
}

//メタデータが検索語と重なったかどうかを判定し、少しでも重なった場合は完全に重ねる。抜け出した場合
function piledCheck(metaSet:MetaSet):MetaSet {
		//メタデータが検索語に少しでも重なったかどうか
		if(searchTerritory.piledCheck(metaSet)) {
		//少しでもメタデータが検索語に重なったとき
			//メタデータを完全に重ねる
			metaSet = searchTerritory.pileCompletely(metaSet);
			if(!searchTerritory.checkPiledList(metaSet)) {
			//既に重なっていたメタデータでないならば
				//重なってるメタデータのリストに追加
				addToPiledList(metaSet);
			}
		} else if(!searchTerritory.piledCheck(metaSet) && searchTerritory.checkPiledList(metaSet)) {
		//メタデータが検索語の範囲から離脱したとき
			if(metaSet.getPiledFlag()) {
			//ドラッグ開始時にメタデータが検索語に重なっていたとき
				//重なっているメタデータのリストから削除
				deleteFromPiledList(metaSet);
			}
		}

		return metaSet;
}

function onlyPiledCheck(metaSet:MetaSet):Boolean {
		return searchTerritory.piledCheck(metaSet);
}

//”検索語に重なっているメタデータのリスト”に、引数で得たムービークリップが含まれているかどうか判定
る
function checkPiledList(metaSet:MetaSet):Boolean {
		return searchTerritory.checkPiledList(metaSet);
```

```
}

function addToPiledList(metaSet:MetaSet):Void {
	var list:Array = searchTerritory.getMetaList();
	searchTerritory.deleteExtractButton();
	metaSet.handleRotation(list.length);
	//重なってるメタデータリストを更新する
	searchTerritory.updatePiledList(metaSet, "add");
	getSpell(textbox.text);
}

function deleteFromPiledList(metaSet:MetaSet):Void {
	searchTerritory.deleteExtractButton();
	var list:Array = searchTerritory.getMetaList();
	var myOrder:Number = searchTerritory.myListNum(metaSet);
	//trace("メタリスト中の数::::" + list.length);
	//trace("抜いた番目-1::::" + myOrder);
	list[myOrder].handleReset();
	for(var k:Number = myOrder + 1; k < list.length; k++) {
		//trace("動きます::::" + list[k].getMetaName());
		list[k].handleAdjust();
	}
	searchTerritory.updatePiledList(metaSet, "delete");
	getSpell(textbox.text);
}

function removeMetaSet(metaSet:MetaSet):Void {
	if(metaTerritory.isDraggedFromMetaTerritory(metaSet)) {
		metaTerritory.removeMetaSet(metaSet);
		searchTerritory.addToMetaInSearchTerritory(metaSet);
		searchTerritory.updateMetaInSearchTerritory();
	}
}

function containsMeta(metaSet:MetaSet):Boolean {
	//var list:Array = searchTerritory.getMetaList();
```

```
		var list:Array = searchTerritory.getMetaInSearchTerritory();//一行上と取替えました
		//for(var k:Number = 0; k < list.length; k++) {
		for(var k:Number = 0; k < list.length; k++) {//一行上と取替えました
			if(list[k].getMetaName() == metaSet.getMetaName() && list[k].getMetaKind() ==
metaSet.getMetaKind()) {
				return true;
			}
		}

		return false;
}

//#############################################################################################

/*サーバー*/
var xmlReceptor:XML = new XML();//データの受け皿
xmlReceptor.ignoreWhite = true;//空白を無視
var i:Number;//ループ用

var parseResult:Result;
var metaTerritory:MetaTerritory;

var dataArray:Array = new Array();//重なっているメタデータのリスト

var sender:LoadVars = new LoadVars;//送信するデータ

function getSpell(searchword:String):Void {
		var spell:String = "";

		dataArray = searchTerritory.getMetaList();

		var cast:String = "";
		for(i = 0; i < dataArray.length; i++) {
			if(dataArray[i].getMetaKind() == "Person") {
				cast = "uima:peopleEntities:";
			} else if(dataArray[i].getMetaKind() == "Place") {
```

```
                        cast = "uima:locationEntities:";
                } else if(dataArray[i].getMetaKind() == "Time") {
                        cast = "uima:timeEntities:";
                }
                cast += dataArray[i].getMetaName();

                spell += cast;

                if(i < dataArray.length - 1) {
                        spell += ",";
                }
        }

        main(spell, searchword);
}



function main(spell:String, searchword:String):Void {
        xmlGetter(spell, searchword);
        xmlReceptor.onLoad = function() {
                parse();
                searchTerritory = new SearchTerritory(searchword);
                if(metaTerritory != null && !newSearch) {
                        //trace("メタデータ表示");
                        searchTerritory.makeExtractButton();//メタデータ抽出ボタンの設置
                }
                newSearch = false;
        }
}

function xmlGetter(spell:String, searchword:String):Void {
        var preMeta:String;
        preMeta = sender.checkedList;

        sender.queryString = searchword;
```

```
		sender.checkedList = spell;

		if(preMeta != sender.checkedList || newSearch) {
			if(newSearch) {
				sender.checkedList = "";
			}

			sender.command = "search";

			if(sender.command == "search") {
				sender.send("search.do", "output", "POST");
			}

			sender.command = "relation";

			if(sender.command == "relation") {
				sender.sendAndLoad("search.do", xmlReceptor, "POST");
			}
		}
}

function parse() {
		rootNode = xmlReceptor.firstChild;
		typeNode = rootNode.firstChild;

		var metaNameList:Array = new Array();
		var metaKindList:Array = new Array();
		var importanceList:Array = new Array();
		var personCount:Number = 1;//人重要度カウント用
		var placeCount:Number = 1;//場所重要度カウント用
		var timeCount:Number = 1;//時間重要度カウント用

		while(typeNode) {
			var metaName:String = typeNode.firstChild.nodeValue;
			var metaKind:String = typeNode.attributes.type;
```

```
                    //リストに追加していく
                    metaNameList.push(metaName);
                    metaKindList.push(metaKind);
                    if(metaKind == "Person") {
                              importanceList.push(personCount);
                              personCount++;
                    } else if(metaKind == "Place") {
                              importanceList.push(placeCount);
                              placeCount++;
                    } else if(metaKind == "Time") {
                              importanceList.push(timeCount);
                              timeCount++;
                    }

                    typeNode = typeNode.nextSibling;
          }

          parseResult = new Result(metaNameList.length, metaNameList, metaKindList, importanceList)

          if(newSearch) {
                    searchTerritory.deleteMetaInSearchTerritory();
                    metaTerritory.deleteAllMetaList();
                    searchTerritory.deleteExtractButton();
                    makeNewMetaTerritory();
                    sender.checkedList = "";
          }
}

function deleteAllMetaList():Void {
          metaTerritory.deleteAllMetaList();
}

function makeNewMetaTerritory():Void {
          metaTerritory = new MetaTerritory(parseResult);
}
```

```
function deleteExtractButton():Void {
        searchTerritory.deleteExtractButton();
```

● MetaSet.as（メタデータの実体を表現するクラス）

```
class MetaSet {
        private var metaName:String;
        private var metaKind:String;
        private var importance:Number;
        private var metaSymbol:MovieClip;
        private var old_x:Number;
        private var old_y:Number;
        private var piledFlag:Boolean;
        private var metaDepth:Number;

        private var handle:MovieClip;
        private var mask:MovieClip;
        private var tag:TextField;
        private var tagFormat:TextFormat;

        private var DEFAULT_META_POSITION_Y:Number = 335;
        private var DEFAULT_HANDLE_DEPTH:Number = 50;
        private var DEFAULT_HANDLE_TEXT_DEPTH:Number = 150;
        private var DEFAULT_MASK_DEPTH = 500;

        function MetaSet(_metaName:String,
_metaKind:String ,_importance:Number, _depth:Number) {
                this.metaName = _metaName;
                this.metaKind = _metaKind;
                this.importance = _importance;
                this.metaDepth = _depth;
                this.metaSymbol.swapDepths(this.metaDepth);
                this.metaSymbol = _root.attachMovie("meta_mc", metaKind +
this.metaDepth, this.metaDepth);//meta_mc は仮
                this.metaSymbol.metaSet = this;
```

```
		this.handle = this.metaSymbol.attachMovie("handle_mc", "handle" +
this.metaDepth, this.DEFAULT_HANDLE_DEPTH + this.metaDepth);

		tagFormat = new TextFormat();
		tag = this.handle.createTextField("tag" + this.metaDepth,
this.DEFAULT_HANDLE_TEXT_DEPTH + this.metaDepth, -80, -10, 80, 20);
		{
			tagFormat.font = "handleFont";
			tagFormat.bold = true;
			tagFormat.color = 0xFFFFFF;
			tagFormat.align = "left";
		}
		tag.embedFonts = true;
		tag.setNewTextFormat(tagFormat);
		tag.text = this.getMetaName();

		this.mask = this.metaSymbol.attachMovie("meta_mc", "mask" +
this.metaDepth, this.DEFAULT_MASK_DEPTH + this.metaDepth);
		this.mask._x = 0;
		this.mask._y = 0;

		//testing code
		setMetaSymbolY(DEFAULT_META_POSITION_Y);

		//シンボルにテキストを付随させる
		makeTextField();

		//実体化したインスタンスをインタラクティブにする
		interactiveMeta();

		setHandle();
	}

	public function setMetaName(metaName:String):Void {
		this.metaName = metaName;
```

```
	}

	public function getMetaName():String {
		return this.metaName;
	}

	public function setMetaKind(metaKind:String):Void {
		this.metaKind = metaKind;
	}

	public function getMetaKind():String {
		return this.metaKind;
	}

	public function setImportance(importance:Number):Void {
		this.importance = importance;
	}

	public function getImportance():Number {
		return this.importance
	}

	public function setMetaSymbol(metaSymbol:MovieClip):Void {
		this.metaSymbol = metaSymbol;
	}

	public function getMetaSymbol():MovieClip {
		return this.metaSymbol;
	}

	public function setOld_x(old_x:Number):Void {
		this.old_x = old_x;
	}

	public function getOld_x():Number {
		return this.old_x;
```

```
	}

	public function setOld_y():Void {
		this.old_y = old_y;
	}

	public function getOld_y():Number {
		return this.old_y;
	}

	public function setPiledFlag(piledFlag):Void {
		this.piledFlag = piledFlag;
	}

	public function getPiledFlag():Boolean {
		return this.piledFlag;
	}

	public function setVisible(visibility:Boolean):Void {
		this.metaSymbol._visible = visibility;
	}

	public function setHandle():Void {
		this.handle._x = 50;
		this.handle._y = 50;
		//ハンドルの初期角度を 45 度にする（初期値は 90 度）
		this.handle._rotation -= 45;
	}

	public function handleRotation(piledNum:Number):Void {
		this.handle._rotation = -45;

		for(var i:Number = 0; i < piledNum; i++) {//piledNum は重ねる直前の時点で重なってるメタデータの数
			this.handle._rotation += 45;
		}
```

```
	}

	public function handleAdjust():Void {
		this.handle._rotation -= 45;
	}

	public function handleReset():Void {
		this.handle._rotation = -45;
	}

	private function setMetaSymbolX(metaX):Void {
		this.metaSymbol._x = metaX;
	}

	private function setMetaSymbolY(metaY):Void {
		this.metaSymbol._y = metaY;
	}

	public function deleteMetaSet():Void {
		this.setVisible(false);
		this.metaSymbol.removeMovieClip();
	}

	private function makeTextField():Void {
		//取っ手につけたテキストフィールドのせいで右に、取っ手テキストフィールドの分だけ右にずれちゃうみたい
		var metaText:TextField = this.mask.createTextField(metaKind + importance + "_txt", metaDepth + 1, 0, 30, this.metaSymbol._width - 80, 20);
		var metaTextFormat:TextFormat = new TextFormat();
		//テキストフォーマット
		{
			metaTextFormat.bold = true;
			metaTextFormat.size = 14;
			metaTextFormat.font = "Arial";
			metaTextFormat.color = 0xFFFFFF;
			metaTextFormat.align = "center";
```

```
		}
		metaText.setNewTextFormat(metaTextFormat);
		metaText.text = metaName;
		metaText.selectable = false;
		metaText.autoSize = "center";
		metaText.wordWrap = true;
	}

	function interactiveMeta():Void {
		Mouse.addListener(this.metaSymbol);

		this.metaSymbol.onPress = function() {
			this.metaSet.old_x = this._x;
			this.metaSet.old_y = this._y;
			this.metaSet.piledFlag = _root.onlyPiledCheck(this.metaSet);

			this.onRelease = function() {


				delete this.onMouseMove;
			}

			this.onMouseMove = function() {
				startDrag(this, false);

				this.onRelease = this.onReleaseOutside = function() {
					//if(this._x < -50 || this._x > 450 || this._y
< -50 || this._y > 450) {
					if((this._x < -50 || this._x > 450 || this._y <
-50 || this._y > 450) && this._y < 470) {//#####testing
					//座標修正をする
						this._x =
_root.dropCheckX(this.metaSet, this._x);
						this._y =
_root.dropCheckY(this.metaSet, this._y);
					} else if(this._y <
```

```
this.metaSet.DEFAULT_META_POSITION_Y - 5 && this._y >= -50) {
                                        //検索領域にメタデータが落とされたら

        _root.removeMetaSet(this.metaSet);
                                        } else if(this._y >= 470)
{//##############################################testing
                                        //メタデータが画面下に出された場合
                                                this.metaSet.deleteMetaSet();//出さ
れたメタデータを削除する
                                        }

                                        //メタデータが検索語に重なったかどうかを
判定し、重なっていた場合”検索語に重なっているメタデータのリスト”に登録する
                                        this.metaSet =
_root.piledCheck(this.metaSet);

                                        //ドラッグをやめる
                                        stopDrag();
                                }

                                delete this.onMouseMove;
                        }
                }
        }

}
```

● MetaTerritory.as（メタデータ領域を表現するクラス）

```
class MetaTerritory {

        //メタ領域上の全てのメタデータリスト（人、場所、時）
        private var allMetaList:Array;
        //メタ領域上で現在表示しているメタデータリスト
        private var showingMetaList:Array;
```

```
		//新たに抽出されたメタデータリスト
		private var newMetaList:Array;

		/**
		  * コンストラクタ
		  */
		function MetaTerritory(result:Result){
			createNewMetaList(result);
			changeForNewMetaList();
		}

		/**
		  * 新たに抽出されたメタデータを作成する
		  */
		public function createNewMetaList(result:Result):Void {
			newMetaList = new Array();

			for(var i:Number = 0; i < result.getNumber(); i++) {
				newMetaList[i] = new MetaSet(result.getMetaName(i),
result.getMetaKind(i), result.getImportance(i), _root.order);
				_root.order++;
			}
		}

		/**
		  * 新たに抽出されたメタデータに入れ替える
		  */
		public function changeForNewMetaList():Void{
			if(newMetaList != null){
				//新しく抽出されたメタデータのリストが空じゃないなら、
allMetaList の中身に突っ込む
				allMetaList = newMetaList;
				newMetaList = null;
				changeMetaKind("Person");
			}
		}
```

```
/**
  * 表示されるメタデータの種類を入れ替える
  */
public function changeMetaKind(metaKind:String):Void{
        showingMetaList = new Array();
        for(var i:Number = 0; i < allMetaList.length; i++){
                if(allMetaList[i].getMetaKind() == metaKind){
                        if(!_root.containsMeta(allMetaList[i])) {
                                showingMetaList.push(allMetaList[i]);
                        }
                }
        }
        updateMetaList();
        for(var j:Number = 0; j < showingMetaList.length; j++) {
                showingMetaList[j].setMetaSymbolX(20 + 120 * j);
        }
}

/**
  * メタデータ領域からドラッグされたか調べる
  */
public function isDraggedFromMetaTerritory(metaSet:MetaSet):Boolean{
        for(var i:Number = 0; i < showingMetaList.length; i++){
                if(metaSet == showingMetaList[i]){
                        return true;
                }
        }
        return false;
}

/**
  * メタデータ集合をはずす
  */
public function removeMetaSet(metaSet:MetaSet):Void{
        var tempArray:Array = new Array();
```

```
		for(var i = 0; i < allMetaList.length; i++){
			if(metaSet != allMetaList[i]){
				tempArray.push(allMetaList[i]);
			} else{
				allMetaList[i].setVisible(false);
			}
		}
		allMetaList = tempArray;

		tempArray = new Array();
		for(var i = 0; i < showingMetaList.length; i++){
			if(metaSet != showingMetaList[i]){
				tempArray.push(showingMetaList[i]);
			}
		}
		showingMetaList = tempArray;
		updateMetaList();
	}

	/**
	 * メタデータ集合を全て削除する
	 */
	public function deleteAllMetaList():Void {
		for(var i:Number = 0; i < allMetaList.length; i++){
			allMetaList[i].deleteMetaSet();
		}

		allMetaList.splice(0);
	}

	/**
	 * メタデータリストの表示を更新する
	 * showingMetaList が更新される度に呼ばれる
	 */
	private function updateMetaList():Void{
		for(var i = 0; i < allMetaList.length; i++){
```

```
                        allMetaList[i].setVisible(false);
                }

                for(var i = 0; i < showingMetaList.length; i++){
                        showingMetaList[i].setVisible(true);
                }
        }
}
```

● Result.as（メタデータ抽出結果を管理するクラス）

```
class Result {

        private var number:Number;
        private var metaNameList:Array;
        private var metaKindList:Array;
        private var importanceList:Array;

        function
Result(number:Number,metaNameList:Array,metaKindList:Array,importanceList:Array){
                this.number = number;
                this.metaNameList = metaNameList;
                this.metaKindList = metaKindList;
                this.importanceList = importanceList;
        }

        public function getNumber():Number{
                return number;
        }

        public function getMetaName(index:Number):String{
                return metaNameList[index];
        }

        public function getMetaKind(index:Number):String{
```

```
                return metaKindList[index];
        }

        public function getImportance(index:Number):Number{
                return importanceList[index];
        }
}
```

● SearchTerritory.as

```
class SearchTerritory {
        private var searchSymbol:MovieClip;//検索語
        private var extractButtonSymbol:MovieClip;//抽出ボタン
        private var metaInSearchTerritory:Array = new Array();//検索領域内にあるメタデータのリスト
        private var metaList:Array = new Array();//検索語に重なっているメタデータのリスト

        var META_WIDTH:Number = 100;//検索語の幅（直径）
        var SEARCH_WIDTH:Number = 100;//検索語の高さ（直径）

        //コンストラクタ
        function SearchTerritory(searchWord:String) {
                makeSearchSet(searchWord);
        }

        //getter/setter
        public function setSearchSymbol(searchSymbol:MovieClip):Void {
                this.searchSymbol = searchSymbol;
        }

        public function getSearchSymbol():MovieClip {
                return searchSymbol;
        }
```

```
public function setMetaList(metaList:Array):Void {
	this.metaList = metaList;
}

public function setVisible(visibility:Boolean):Void {
	for(var j:Number = 0; j < metaInSearchTerritory.length; j++) {
		metaInSearchTerritory[j].deleteMetaSet();
	}
}

public function getMetaList():Array {
	return metaList;
}

public function getMetaInSearchTerritory():Array {
	return metaInSearchTerritory;
}

public function updateMetaInSearchTerritory():Void {
	for(var j:Number = 0; j < metaInSearchTerritory.length; j++) {
		metaInSearchTerritory[j].setVisible(true);
	}
}

/**
  * 検索語を画面に表示させる
  */
private function makeSearchSet(searchWord:String):Void {
	searchSymbol = _root.attachMovie("search_mc", "search_mc", 1000);
	//検索語位置
	{
		searchSymbol._x = 200;
		searchSymbol._y = 100;
	}

	var searchText:TextField;
```

```
                searchText = searchSymbol.createTextField("search_txt", 1001, 0, 30,
searchSymbol._width, 20);
                var searchTextFormat:TextFormat = new TextFormat();
                //テキストフォーム
                {
                        searchTextFormat.bold = true;
                        searchTextFormat.size = 16;
                        searchTextFormat.font = "Arial";
                        searchTextFormat.color = 0x000000;
                        searchTextFormat.align = "center";
                }

                searchText.setNewTextFormat(searchTextFormat);
                searchText.text = searchWord;
                searchText.selectable = false;
                searchText.autoSize = "center";
                searchText.wordWrap = true;
        }

        /**
          * 抽出ボタンを生成する
          */
        public function makeExtractButton():Void {
                this.extractButtonSymbol  =  _root.attachMovie("extractButton_mc",
"extractButton_mc", 5000);
                //告知位置
                {
                        this.extractButtonSymbol._x = 375;
                        this.extractButtonSymbol._y = 50;
                }

                interactiveInfo();
        }

        /**
          * 抽出ボタンを押せる様にする（インタラクティブにする）
```

```
  */
public function interactiveInfo():Void {
        Mouse.addListener(this.extractButtonSymbol);

        this.extractButtonSymbol.onPress = function() {
                //抽出ボタンがクリックされたとき
                this.onRelease = function() {
                        _root.deleteExtractButton();
                        _root.deleteAllMetaList();
                        _root.makeNewMetaTerritory();

                        delete this.onMouseMove;
                }
        }
}

/**
  * 抽出ボタンを不可視にする（押せなくする）
  */
public function deleteExtractButton():Void {
        //this.extractButtonSymbol.removeMovieClip();
        this.extractButtonSymbol._visible = false;
}

/**
  * 引数で得たメタデータを検索領域にあるメタデータのリストに追加する
  */
public function addToMetaInSearchTerritory(metaSet:MetaSet):Void {
        metaInSearchTerritory.push(metaSet);
}

/**
  * メタデータの y 座標を old_x（ドラッグする直前の x 座標）にする
  */
public function dropCheckX(metaSet:MetaSet, metax:Number):Number {
        metax = metaSet.getOld_x();
```

```
		return metax;
	}

	/**
	 * メタデータの y 座標を old_y（ドラッグする直前の y 座標）にする
	 */
	public function dropCheckY(metaSet:MetaSet, metay:Number):Number {
		metay = metaSet.getOld_y();

		return metay;
	}

	/**
	 * メタデータを検索語に完全に重ねる（座標を同値にする）
	 */
	public function pileCompletely(metaSet:MetaSet):MetaSet {
		var metaSymbol:MovieClip = metaSet.getMetaSymbol();

		var distanceX:Number = searchSymbol._y - metaSymbol._y;
		var distanceY:Number = searchSymbol._x - metaSymbol._x;

		//座標を揃える
		metaSymbol._x = searchSymbol._x;
		metaSymbol._y = searchSymbol._y;

		return metaSet;
	}

	/**
	 * メタデータが検索語に重なったかどうかを判定する
	 */
	public function piledCheck(metaSet:MetaSet):Boolean {
		var metaSymbol:MovieClip = metaSet.getMetaSymbol();

		var distance = Math.sqrt(Math.pow((metaSymbol._x -
```

```
this.searchSymbol._x), 2) + Math.pow((metaSymbol._y - this.searchSymbol._y), 2));//引数で得たメタデータの中心と、検索語の中心との距離

                if(distance < META_WIDTH/2 + SEARCH_WIDTH/2) {
                //少しでも重なっている場合
                        return true;
                }

                return false;
        }

        /**
          * 引数で得たメタデータが、”検索語に重なってるメタデータのリスト”に含まれているかどうかを調べる
          */
        public function checkPiledList(metaSet:MetaSet):Boolean {
                var i:Number;
                for(i = 0; i < metaList.length; i++) {
                        if(metaList[i] == metaSet) {
                                return true;
                        }
                }

                return false;
        }

        /**
          * 検索語に重なってあるメタデータのリストを更新する（追加または削除）
          */
        public function updatePiledList(metaSet:MetaSet, command:String):Void {
                if(command == "add") {
                        if(pushAdmission(metaSet)) {
                                metaList.push(metaSet);
                        }

                } else if(command == "delete") {
```

```
                        var index:Number;
                        for(index = 0; index < metaList.length; index++) {
                                if(metaList[index] == metaSet) {
                                        break;
                                }
                        }
                        metaList.splice(index, 1);
                }
        }

        /**
          * 引数のメタセットがメタリストの何番目なのかを返す
          */
        public function myListNum(metaSet:MetaSet):Number {
                var i:Number;

                for(i = 0; i < metaList.length; i++) {
                        if(metaSet == metaList[i]) {
                                break;
                        }
                }

                return i;
        }

        /**
          * 引数のメタデータが既にメタリストに登録されているかどうかを判定する
          */
        private function pushAdmission(metaSet:MetaSet):Boolean {
                var admission:Boolean = true;

                for(var i:Number = 0; i < metaList.length; i++) {
                        if(metaList[i].getMetaName()  ==  metaSet.getMetaName()
&& metaList[i].getMetaKind() == metaSet.getMetaKind()) {
                                admission = false;
                                break;
```

```
                        }
                        admission = true;
                }

                return admission;
        }

        /**
          * 検索領域にあるメタデータを全て差う所する
          */
        public function deleteMetaInSearchTerritory():Void {
                for(var i:Number = 0; i < metaInSearchTerritory.length; i++) {
                        metaInSearchTerritory[i].deleteMetaSet();
                }

                this.metaList.splice(0);
                this.metaInSearchTerritory.splice(0);
        }

}
```

3.2. サーバサイド実装

3.2.1. 開発環境

Eclipse SDK 3.1

Tomcat v5.5 Server

3.2.2. 実装項目

Java Struts サーブレット

XML 出力 JSP

3.2.3. 実装手順

DMC が IBM のサンプルコードを元に作成した、現状で利用しているアプリケーションの

ソースコードを提供して頂き、それを解析して、クライアント側と HTTP でやり取りするプログラムを作成した。元のソースコードへの変更点は以下の通り。

- 検索結果の HTML を新 UI に適した状態で出力する
- 「メタデータ抽出」要求に対しては関連語を XML データで返す
- 「人」「場所」以外に「時」の関連語も取得する設定を付加
- Flash からカンマ区切りで送られてきた関連語を配列にセットし直す
- 検索結果と関連語が順番に別のセッションでリクエストされるので、関連語を取得する前に検索結果を再度取得する必要がある
- XML で送信する値をフォームに直接セットするコード（初期統合テスト時）
- 検索結果オブジェクトを作成する API のスタブを作成（後期統合テスト時）

3.2.4. ソースコード

●SearchAction.java（プロパティ・設定の変更や削除部分を除いた、スタブ関係以外の追加コード）

・変更メソッド

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
```

・変更部分１

```
// csv で送られてきた checkedList を配列に変更する追加コード
String[] checkedListTemp = (String[])
theForm.get(SearchDynaValidatorForm.PROPERTY_CHECKED_LIST);
String csvString;
if(checkedListTemp.length != 0){
        csvString = checkedListTemp[0];
        checkedListTemp = csvString.split(",");

theForm.set(SearchDynaValidatorForm.PROPERTY_CHECKED_LIST,checkedListTemp);
}
```

・変更部分２

```
} else if (command.equals(SearchDynaValidatorForm.COMMAND_RELATION)) {
    // 関連語を取得する前に検索結果を取得する追加コード
forward = processQueryAction(mapping, form, errors,
```

```
request,SearchDynaValidatorForm.PAGE_RESULTS);
forward = relationAction(mapping, form, errors, request,
SearchDynaValidatorForm.PAGE_RELATION);
}
```

●metaDataXML.jsp（関連語をクライアント側に XML で送り返す）

```
<%@ page contentType="text/xml;charset=utf-8"%>
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-logic" prefix="logic" %>

<?xml version="1.0" encoding="utf-8"?>
        <metadata>
        <logic:equal name="relationExists" value="true" scope="request">

          <bean:define id="peopleEntityList" name="SearchForm"
property="peopleEntities"/>
          <bean:size id="peopleListSize" name="peopleEntityList"/>
            <logic:greaterThan name="peopleListSize" value="0">
              <logic:iterate id="people" name="peopleEntityList">
                <logic:present name="people">
                  <metaword type="Person"><bean:write name="people"/></metaword>
                </logic:present>
              </logic:iterate>
            </logic:greaterThan>

          <bean:define id="locationEntityList" name="SearchForm"
property="locationEntities"/>
          <bean:size id="locationListSize" name="locationEntityList"/>
            <logic:greaterThan name="locationListSize" value="0">
              <logic:iterate id="location" name="locationEntityList">
                <logic:present name="location">
                  <metaword type="Place"><bean:write name="location"/></metaword>
                </logic:present>
              </logic:iterate>
            </logic:greaterThan>
```

```
        <bean:define id="timeEntityList" name="SearchForm"
property="timeEntities"/>
        <bean:size id="timeListSize" name="timeEntityList"/>
          <logic:greaterThan name="timeListSize" value="0">
            <logic:iterate id="time" name="timeEntityList">
              <logic:present name="time">
                <metaword type="Time"><bean:write name="time"/></metaword>
              </logic:present>
            </logic:iterate>
          </logic:greaterThan>


        </logic:equal>
        </metadata>

        <logic:notEqual name="relationExists" value="true" scope="request">
        </logic:notEqual>
```

3.3. 統合テスト

3.3.1. クライアントサイドとサーバサイドの結合テスト

実施期間：7 月 6 日～7 月 2１日
実施環境：Tomcat を起動させた Eclipse 上

リクエスト・レスポンスの送受信や XML の受け渡しに始まって、最終的には検索用 API のスタブを作成して検索結果・関連語を出力するまで。

Flash からは、HTML のチェックボックス型 input フォームのように、同じ名称の引数を複数送信してサーバ側で配列として受け取ることは不可能だと判明した。カンマで区切り１つのデータとして送信し、サーバ側で配列としてセットし直すように変更した。

3.3.2. DMC の環境における統合テスト

実施期間：7 月 28 日 10 時 30 分～14 時 30 分
実施環境：DMC サーバー上

ユーザーテスト実施直前のデプロイ作業。デプロイ自体は PM のサポートのもとに滞りなく遂行。スタブ環境では発見できなかった、クライアント側アプリケーションのバグを主に修正した。加えて検索結果の出力画面のレイアウトも調整した。

修正案件１：検索語に重ねた人名関連語が検索結果に反映されない
原因：クライアント側の送信引数に付加する文字列の誤り

修正案件２：検索語に重ねた地名関連語が検索結果に反映されない
原因：クライアント側の送信引数を処理する条件分岐

4. 評価

4.1. 場所

慶應義塾大学西別館 2F（デジタルメディア・コンテンツ統合研究機構）

4.2. 日時

2006 年 7 月 28 日 15 時 20 分 ～ 16 時 15 分

4.3. 使ってもらった様子

好評だったようだ。
テストのはじめに各人自由に使って貰った。テスターの方々にはとても積極的にシステムを使って頂けて、その中でもテスター同士の談笑が絶えなかった。
最後に“ 弄っていて楽しい ”、“ 虫眼鏡が引っ張られて勝手に重ねられる様子が面白い ”、“ 検索システムとして、足していく操作だけではなく、引いていく操作も可能である点が良い ”、“ デザインがかわいい ”、“ 面白いものを見せて貰った ” など、様々な感想を頂いた・

4.4. アンケート内容

次の 2 ページは、ユーザー試験時に行ったアンケート内容である。

2006 年 7 月 28 日

DDSS ヴィジュアル検索システム

◆ヴィジュアル検索システム概要

DMC の従来の検索システムのユーザーインターフェースは、初心者にとっては操作がわかりづらい部分があります。特にセマンティック検索では、ユーザーが結果を得るまでのプロセスが複雑で、検索の仕組みを理解することは難しいと思います。今回のヴィジュアル検索システムは、初心者でも簡単に DMC のセマンティック検索が利用可能となるような UI を目指しています。

◆ユーザーテスト概要

DDSS ビジュアル検索が、DMC の長所を理解出来る・感じ取れるシステムであるか、初心者でも使えるインターフェースであるかを確かめるテストです。

◆　　事前調査項目

・ 自身のパソコンに対する習熟度を以下から選んで下さい。

□ 初級者（趣味で利用している程度）

□ 中級者（仕事・学習・研究の道具として利用している）

□ 上級者（ソフトウェア開発などの IT 技術系の仕事・研究に携わっている）

・　　あなたが普段使っている検索エンジンを教えてください。

・ DMC の検索システムを以前に使ったことがありますか？

　はい　　いいえ

◆レビュー項目

・ 虫眼鏡を検索語に重ねて検索結果を得ることができましたか？

　はい　　いいえ

・ 新たな関連語の抽出を行うことができましたか？

　はい　　いいえ

・ 虫眼鏡の上にさらに虫眼鏡を重ねて検索をすることができましたか？

　はい　　いいえ

・ 関連語属性変更ボタンを押すことができましたか？

　はい　　いいえ

・ 抽出された関連語から何か新しい発見を得ることがありましたか？

　はい　　いいえ

・ 検索結果から何か新しい発見を得ることがありましたか？

はい　　いいえ

・ この検索システムをあなただったらどのように使いますか？

・　　この検索システムでの「虫眼鏡」は「メタデータ」を示しています。
この検索システムを使ってみて、「メタデータ」とは何だと思いましたか？

・　　このシステムについて、率直な感想・意見等、ご自由にお書き下さい。

ご協力有難う御座いました。

4.5. アンケート結果

アンケートは『操作』について問う項目と『理解』について問う項目に分かれていて、それぞれ 4 つ、2 つの質問をした。被験者は 6 人で、開発経験者 1 人、WEB デザイナー1 人、人文社会系 4 人で構成されていた。
目標は 50%以上が“ はい ”（合格）と答えることとした。
以下は質問内容と“ はい ”と回答された人の数である。

【操作】

1．　虫眼鏡を検索語に重ねて検索結果を得ることが出来ましたか？　5 人
2．　新たな関連語の抽出を行うことが出来ましたか？　5 人
3．　虫眼鏡の上にさらに虫眼鏡を重ねて検索することが出来ましたか？6 人
4．　関連語属性変更ボタンを押すことが出来ましたか？　4 人

【理解】

1．　抽出された関連語から何か新しい発見を得ることがありましたか？4 人
2．　検索結果から何か新しい発見を得ることがありましたか？　2 人

操作については 24 点満点の 20 点（83.3%）で非常に高い点数結果になった。対して理解については 12 点満点で 6 点（50%）で、目標の値ギリギリであった。

4.6. ユーザーからの要望

感想を聞く中で、いくつかユーザーから「こうしたらより良い」という要望があった。

- 虫眼鏡の柄の色と、検索結果の検索語の色づけを対応させると分かり易い。
- 人と場所のメタデータの色を変えた方が分かり易い。
- 検索語とメタデータを重ねたとき、それぞれの色が混ざった方が良い。
- メタデータを入力バーにドロップしたときに、検索語がメタデータに変わると検索がし易い。
- メタデータが表示されたとき、相関関係が検索語からの距離で表現されていると良い。
- 抽出されたメタデータの意味が分からなかった場合、メタデータにカーソルを合わせると、そのメタデータの意味が表示されると良い。
- Flash 画面と検索結果画面がフレーム分けではなくて、別ウィンドウであった方がより大きく画面が使えると思う。
- 縦長ではなくて横長にした方がマウス操作がし易いのではないだろうか。
- メタデータを抽出すると、メタデータの表示が、既に検索語と重なった状態で表示するという方法もある。

- “移動させて検索”ではなく、ただクリックで検索出来た方が良い。移動させる意味が分からない。
- ズームなどのエフェクトを用いて、絞っていっている、という感覚を表現出来ると良い。
- メタデータ同士の検索が出来ると良い。
- ボタンとか虫眼鏡にもっと“らしさ”を出すと良い。

4.7. 評価の考察

予想以上に操作がスムーズに行われていた。操作がスムーズに出来ていたという事実を、ユーザーの回答で 80%以上の得点を出したことが証明してくれている。逆に理解面の評価については時間・システム的な制限があるということも関係してくるが、まだ改善の余地は十二分にある。色を変える・色を混ぜるといったユーザビリティの向上を目指した機能や、さらに相関距離などといった理解を深める為の機能などを付加することによって、よりユーザーの満足度を高めることが出来るのではないかと、考えた。また、虫眼鏡をもっと“らしく”した方が良いなど、デザイン面も重視されていたので、ユーザーにより“らしく”見えるデザインに変更することで、ユーザーにとって“色眼鏡で見ている”というコンセプトが伝わり易いものになると考えられる。

5. 個人レポート

5.1. 個人レポート（安藤亮一）

１．プロジェクトでの作業に関して

今回、プロジェクトの作業を通して強く実感したことは、プロジェクトの依頼主であるクライアントとの意思の疎通が、非常に難しいということだった。

もともとこのプロジェクトの提案をなさった嶋津先生の話では、「現状の DMC の検索システムの UI が使いづらいものであるため、新しい UI に改善してほしい」というものであった。その話を受けて私たちが思ったことは、現状の UI で問題がある点を挙げ、その部分に関してユーザビリティの向上を目指し、機能面での追加はないというもので、その考えからまず現状の問題点を挙げ、それらの改善と難しい用語の再定義、チュートリアルの作成という改善を行う計画を立てた。だが、クライアントとしての要求は、「何か使いやすくて新しい、魅力的な UI を作成してほしい」というもので、私達が理解していたのとはちがうものであった。この点でクライアントと PM を含めたプロジェクトメンバーの両者の理解が食い違っていたため、そこからメンバーの認識を一致させるのに再びの努力が必要となり、少しスケジュールが詰まることとなった。

前回のプロジェクトもクライアントからの要求をもとに何かを提案するという形だったが、そのクライアントの一人である社員が PM でもあったため、認識がずれていた場合そのつど指摘し、正しい方向に導いてくれたため、認識のずれはあまり大きくなることはなかった。だが、今回の場合、実際に会えるのは週に 1 回わずかな時間で、メールでのやり取りが主だったため、認識のずれが大きくなることが多々あった。この点は、前のプロジェクト以上に苦労した点であった。

２．プロジェクトのスケジュール管理に関して

前回のプロジェクトで、実装がかなり後のほうまでずれ込み、実装・テストが大変になった経験があったので、今回はきっちりとしたスケジュール管理の下、計画立ててプロジェクトをすすもうと考えていた。だが、実際にふたを開けてみると、前のプロジェクト以上にきついスケジュールで、最終成果物がかなりぎりぎりの仕上がりとなった。

原因としては、クライアントの要件があいまいのままプロジェクトがすすんでいったことであると思う。嶋津先生から、「こういったものがほしい」という要望はあったが、そこから要件を考えることをせず、漠然と要望をそのまま受け入れてプロジェクトを進めていったため、方向が定まらなかった。7 月に入りこのままじゃまずいとプロジェクト内でも認識して、要件のまとめなおしを行ったところ、仕様を変更することとなった。この時点での仕様の変更はスケジュール的に非常に厳しいものであったが、実際に要件をまとめなおしたことでプロジェクト内での認識も統一され、プロジェクトが目指すものが明らかにな

ったため、評価方法なども定まることとなった。

前回のプロジェクトにおいても、要件定義に多くの時間を割いてスケジュールがきつくなったのだが、今回も要件を明らかにしないまま進めてスケジュールを遅らせてしまったので、同じ失敗を繰り返してしまったことは後悔する点である。

３．PM に関して

前回と今回のプロジェクトで、一番大きく差があると感じた点は PM である。

前回の PM であった佐々木さんは、何か作業が終わるたびに次の作業の指示をして、常に舵取りをしながらメンバーを引っ張っていくやり方であったが、今回の PM である竹元さんは、メンバーが自ら進んでいくのを促し、わからない点などについてアドバイスをしながら進めていくやり方であった。自分としては最初は非常に異なるマネジメントだったため、違和感があったのだが、プロジェクトを進めていくうちに、進め方は PM によって異なるもので、それぞれのコンセプトを持って進めているのだなということに気づいた。

自分のやりやすさとしては、佐々木さんのようにぐいぐいと引っ張ってくれるやり方だと、迷わずに進めることができてやりやすかったのだが、今回のように進め方を任されて自分で進んでいくことで、ある種の責任感のようなものを覚えて、色々と思案する機会が増え、今までにない経験をすることができた。

プロジェクトの進め方は PM 次第で大きく変わってくるというのは、自分にとっては、かなり大きな発見であった。

5.2. 個人レポート（川口将司）

今期の研究会とプロジェクトを振り返ってみようと思う。

今期の研究会は第一回目で PM の方々と面談をして自分が参加したいプロジェクトを選定し、逆に PM の方々に選んで貰うという、これは縮小版就職活動という感じだった。PM の人柄やプロジェクト内容から、私は第一志望に検索システムの改善プロジェクト（後の DDSS）を挙げた。どうやら竹元さんに選んで頂けたらしい（もしくはお零れだったかもしれないが)。このプロジェクトを選んだ正直な理由を詳しく述べると、PM と会話をしていて人柄が良かったのも本当だが、後はプロジェクト自体が比較的分かり易くってやるべきことがはっきりとしていそう・・・つまり簡単そうだったからである。しかしこう思っていた自分を後に後悔することになる。だがプロジェクトを選んだことについては後悔していない。

プロジェクト開始当初、私（達）は DMC の検索システムのサイトを改善するだけのプロジェクトだと思っていた。言葉の意味が判りづらい、リンクの配置が気づき辛いなど、既存のサイトに難癖をつけていた。しかし後々になってクライアントである嶋津さんの要望は、それまでやってきた内容とは全く違ったものであることに気づくこととなる。私は大げさではなく、プロジェクトが転覆したと思った。その時点で既に 5 月 11 日だった。

すぐにプロジェクトの方向性を変更することは難しかった。突然 “ 検索システムに新規の機能を追加して欲しい ” と言われても、早々考え付くものではない。嶋津さんから聞いた例を参考にしたところ、メンバー（荒木さん）から出ていた案 “ ベン図を検索の操作にしたら面白そう ”が利用出来ると考えた。ミーティング時に議論を行い、煮詰めていった。一時は仕様の概要が決まった。その際、開発言語についても議論した。候補としては Java,ajax,Flash の 3 種類が挙げられたが、実行環境の普及具合や（この辺には誤解があったかもしれない・・・JavaVM の方が普及している様だ)、私に Flash の知識が多少あったことから Flash に決定した。

また、クライアントサイド／サーバーサイドで 2 人ずつ担当を分けることになった。後にこの担当分けの意味が薄れていく。

次に Flash の学習に入った。Flash／ActionScript は多少弄った経験があるのだが、今回の実装については、とてもではないが中途半端な知識では実装出来そうになかった。メンバー（安藤君）と二人で技術調査などを行う。変数のスコープが非常に煩わしく、かなり苦戦した。クライアントとサーバーで XML のやり取りをすることは決めていたので、やり取りをするだけの部分の機能を作ってみようとしたが、なかなか上手く行かなかった。結局この機能も 7 月の上旬に実装に使える形として完成することになる。

とりあえずの仕様が決定した後、次の週、そのまた次の週となってくると、段々『こっちの方が良い！』と思える様な部分部分の仕様が出てきて、仕様がなかなかはっきりとは決まらなかった。この仕様の迷走は後に数週間続き、結局仕様が決定したのは 7 月の上旬辺りであった。この辺りでサーバーはほぼ元々あったソースを流用出来ることが判明し、サーバー側は 1 人でよくなった。

仕様が決定した時には、メンバーである学生達はレポートや試験で忙しくなる時期に入ってしまい、PM の竹元さんが DMC へ赴き、仕様決定の報告を行うこととなってしまった。本来ならばメンバーも直接赴きヒアリングすべきなのだが、仕様の決定がどんどん遅れていってしまった為、結局メンバーがやるべき作業を PM にやって貰うことになってしまった。

7 月中旬に入り、メンバー（安藤君）と一気にクライアントの設計をし、そのまま実装に入った。その週のうちに実装を終え、クライアント単体のテストをし、バグフィックスを行い、デプロイの申し込みにまで漕ぎ着けた。DMC 側の都合のこともあり、デプロイは翌週の金曜日となった。予定よりかなり厳しいスケジューリングとなった。

金曜日の午前 9 時過ぎ頃 DMC へ到着し、早速デプロイ作業を行った。デプロイするまでが大変だったが、デプロイした後に発覚したバグの数も眩暈がした。今までがスタブを積んだ（仮）サーブレットとクライアントでのテストだった為、どうしても結合時に多くのバグが発生したのだ。バグは全部で 5,6 個程出て、3,4 箇所の機能を直すのに 5 時間程度の時間を要した。2 箇所は元より想定していなかったバグで、設計にも対策は含まれていなかった上に、その場で修正する為の時間もなかった・・・何より目立ちすぎるバグというわけではなかったので見送った。

後のテスト時には技術職のテスターの方々が“ 当日結合テストを行って、よくちゃんと動く形でデプロイできたね ”とか“ ほとんど一発で動いた様なものではないか ”という言葉をかけて下さった。デプロイ作業を頑張った甲斐があったなあと思った。ちゃんと動いているかどうかは多少のバグに目を瞑れば、ではあるのだが。

テスト中、テスターの方々が楽しそうに使っている姿を見て、大きな達成感を得られた。割としんどい日々が続いていたが、なんとか“ 本当の最終期限 ”までに完成させることが出来て良かったと思えた。評価も操作性についてはとても高い点数を頂けたので満足している。

プロジェクト全体の感想は、とにかく色々しんどかった。色々というのは・・・中盤、プロジェクトのフットワークがとにかく重かったことが一つだ。プロジェクト全体の調子

がやっと上がってきたのは終盤（7 月）に入ってきてからだと思う。仕様がさっさと決まらないとどうしても不安で、不安を残したまま先に進むことは躊躇われるのだ。この不安の源になったのは、メンバー4 人全員がなかなか集まれなかったこと。基本的に僕と安藤君が会って話し合っていたが、次に 4 人が集まれる日となると次回の研究会となってしまう。メールでの情報のやり取りや約束の取り付けが甘かったと言われればそれまでなのだが、それ以前に 4 人が同時に集まらないと効率がガタ落ちてしまう。そこをカバー出来なかったのが一番悪いとこだ。対策としては・・・やはり毎週定時に必ず集まる様にプロジェクト開始当初から決めておくべきだろうか。今回はそれを実行しようとはしたものの、なし崩しで結局取り決めなかったことが悪かった。

7 月に入った後は充実していた（し過ぎる程だった)。7 月に入った後は次の手次の手、どんどんやるべき事が雪崩れ込んで来た為、作業量や時間の割り振りがとても辛かった。バグフィックスにしても、多すぎるバグの前で膝をつきそうにもなった（この辺はコーディング設計が悪かったせいでもあるのだが・・・)。また 7 月下旬になり、テスト当日、結合テストの時間があまりに短かったことで、バグフィックスがテストに間に合うのか、追い込まれた気分になった。

しかし、時間内になんとか使える状態にし、実際にテストを行ってみると、ユーザーの方々から色々好評及び改善についての意見などを頂いたときは本当に達成感があった。その時ばかりは最終ドキュメントのことや最終発表のことも忘れることが出来るくらいだった。

終わり良ければ・・・なんていうと陳腐に感じてしまうが、生きてるうちにそんな経験なんかい味わえるだろうか。そのまま終わりも悪くなってしまうケースも、絶対多いと思う。今回は序盤～中盤の事の運びには課題が残るが、終盤の事の運びはとても良かったと思う。とにかく何にせよプロジェクト大失敗に終わらずに良かった。

以上。

5.3. 個人レポート（瀬端博志）

１．プロジェクトについて

この研究会に参加するのが初めてなだけではなく、PM の下、数人でプロジェクトを行うというのも初体験である。終わり間近になって自分の働きぶりを思い返してみると、割り振られたタスクの中では頑張って作業をこなしていたように思う。ただ、舵を取り仕切ったり、自分から「何々の作業をやります」と宣言したりすることは殆どと云って良いほどなかったので、その部分について努力することができていれば自己採点で百点を付けていただろう。

クライアントである嶋津先生からの要望を、当初は単に「UI 改善」として受け取っていたが、ヒアリングを繰り返して意見のやり取りを進めて行くと、実は私たちの発想力を活かした「新 UI」を望んでいることが判った。その後メンバー間で、DMC の検索エンジンが最も強調するべきである「セマンティック検索」に着眼点を置いて、新 UI の雛形であるベン図検索をアイデアとしてまとめることとなった。

このようにクライアントとの話し合いを通じて意思の疎通をはかり、その後メンバー間で煮詰めるといった作業はとても有益なものだった。このプロセスこそが「依頼されたモノを多人数で作成」する際のキモであり、難しい所でもあるのだろう。

ネット上で目にした SE やプログラマの愚痴の大部分は、ここで躓いて発生してしまったトラブルによるものであるように思える。勿論企業間のやり取りとなると色々な要素が関わってきて、そう簡単に問題を断言できるものでもないだろうが、これは基本的であり、最も重要な要素だという印象を受けた。軌道修正によって遅れが発生した雰囲気が漂ったのは事実だが、そこで何を作るのか、何をするべきなのかがクリアになって、ある程度の達成感と、緊張感が生まれたことを考えれば、その遅れに見合った成果が得られていたのだと判断できる。

中間発表が終わった後にプロジェクトの提案書を作成する課題が持ち上がって、その時に私たちは何を作っているのか？を改めてはっきりさせるための話し合いがあった。この検索システムのターゲットは誰なのか、という点に話が及び、それは DMC やクライアントが考える要素であってベン図 UI の製作者が設定するものなのか、という意見が出た。そこでクライアントの要望に戻って、細かい要求分析をして、それから私たちはこのような UI を作ることにしました、という理屈を明確に把握した時は目から鱗が出たような気分であった。このように強く心に残った経験は、またいつかプロジェクトメンバーとして活動する際に活かせるだろう。

２．PM について

PM の下で働くという経験がないので、PM はどうあるべきか、という話題につい

ては自信がない。個人的な印象としては、竹本さんは普段はメンバーの活動を見守っていて、困った時にサポートにまわったり疑問に答えてくれるという形で頼りがいがあった。ただ、率先して指示を下したり細かなスケジュールを管理することはなく、メンバー間で決定したタスクに期間を設けたり、全体の WBS を作成したりすることが主だったので、稀に「このままでスケジュールに問題はないだろうか」と心配することはあった。これはメンバーの自主性を計って、そのままでも大丈夫だろうと判断されたのだとは思う。ただ私の性格からすると、疎ましく感じない程度に指示が与えられていると気が楽なので、上記のように感じてしまったということはある。

この疑問を解消するには、プロジェクトをこなして他のタイプの PM とも作業経験を積むのが一番であろう。

5.4. 個人レポート（荒木恵）

プロジェクトにおいての活動

私は今学期は、教職の授業と研究会の時間がかぶってしまったため、この研究会は聴講という形で参加した。また、今期は教育実習があったため、３週間はプロジェクトの活動に参加できないことをあらかじめ PM の竹元さんに伝えた。

この DDSS プロジェクトにおいて私は、ヒアリングへの参加、ベン図のアイデアの提案、ミーティングでの議論など、発言の機会は多くいただいた。しかし、実装はほとんど行なわなかった。私が今期の研究会で作ったものは、ユーザテスト用アンケートと、発表資料、議論のまとめなどである。

今期学んだこと

先学期と比較して

- PM について

私が先学期と最も違いを感じたのは、プロジェクトマネージメントの方法である。昨年は、PM が大学院生である明石さんであったこともあり、プロジェクトを進める上で何か分からないことがあれば、すぐに PM に聞くことができた。

しかし、今回の PM である竹元さんは、社会人であり、木曜日にしか会うことができない。木曜日以外に連絡を取りたい場合は、メールに頼るしかなく、細かい質問や相談をすることは難しかった。

また、プロジェクトの進め方にも差があった。

先学期は、PM がミーティングの進行なども管理し、ログを取ることを促し、ミーティングや作業の終了条件を定めるなど、「プロジェクトの中で私は何をすればよいのか？」についての示唆を与えていた。

今学期は、PM ミーティングの時間と、授業後のミーティングの時間がかぶったこともあり、PM 不在でミーティングをすることが多かった。また、今学期は研究室が混んでいたため、私たちは大学構内のサブウェイでミーティングをしていた。そのため、PC を出しながらミーティングするということがなく、議論はすすむものの、あまりきちんとミーティングログを残せないことが多かった。

また、今学期の PM である竹元さんのプロジェクトマネージメントの手法は、「引っ張ってゆく」「指示を与える」というよりは、「サポートする」「質問に応える」というやり方で、「今何をしたらよいのだろう？」という状態になることが何度かあったが、そのたびにメンバー間で議論が起こり、メンバーの考える力は育ったと思う。

- 顧客がいることについて

先学期のプロジェクトと大きく違うところに、顧客の存在がある。今学期のプロジェクト

の顧客である嶋津先生は、私たち学生の度重なるヒアリングにも快く応じてくださり、また、プロジェクトの進行方法についてもアドバイスをしてくださった。
実際にそのシステムを待っている顧客がいる、ということは、大きなモチベーションになった。
また、プロジェクトをすすめるにあたって、「要求分析」「要件定義」という、先学期は無かったフェーズを体験することができた。
私たちのプロジェクトでは、はじめ、UI の改善とは、「今ある UI の色や配置を改良すること」だと考えていたが、嶋津先生への２回目のヒアリングで、「新しい UI,新しい発想」と、セマンティック検索システムの使い方そのものに対するアイデアを求められていることが分かった。、この「求めるものの差」に気づくまでに、プロジェクトの開始から約３週間を要したことから、顧客とプロジェクトメンバーの認識の差というものを知ることができた。
またさらに、顧客の要求が分かってから、それを要件の形にまとめるまでに、さらに２ヶ月を要したこと、とくに、「要求を要件にまとめなければならない」ことを自覚するまでに１ヶ月半近くかかったことから、「要求を理解したところで、何を作ればいいか決まったわけではない」ということを知ることができた。
また、私は同席できなかったが、ユーザーテストの様子をビデオで見て、楽しそうに使ってもらっている様子から、「実際に使ってもらえる」ことの楽しさを知ることができた。

今学期の活動全体を通じて

今学期の活動を通じて、私が学んだ一番大きなことは、まず、PM ごとにプロジェクトのやり方が違うということ、そして、要求を要件に落とし込むところが、プロジェクトの面白いところだ、ということである。
一つ目の、PM ごとにやり方が違う、という事は、その中でのプロジェクトメンバーの動きも違う、ということである。プロジェクトに参加する際には、PM のやり方を受け入れながら、「このプロジェクトの中で自分はどのような役割を果たすべきか」について、メンバーも考える必要性があるのだと感じた。
また、要求を要件に落とし込む、ということに関しては、私たちは要求を聞いてそこからアイデアを出し、すぐに実装に取り掛かってしまったが、要求⇒要件⇒仕様⇒実装という流れをきちんと作らないと、何をもってプロジェクト完了とするか、ということが分からないままになってしまう、ということが分かった。このことは、自分たちで作りたいものを作るよりも、顧客がいたほうが、感じやすかった。