文献サーベイブログシステム 『 Survlog 』

五 嶋 宏 通[†1] 服 部 健 太[†1] 松 本 昌 憲[†1]

通常，研究者は研究活動を行うに当たって過去の多くの文献を調べ，知識の体系化を行う．そのとき，文献自体をどのように保存するかやどの文献を読んだのか，まだ読んでいないのかの管理，読んだ文献の内容メモ，文献の分類分けなどの作業が必要となる．我々はこのような研究における文献調査活動を簡単に行えるようにサポートを行うためのシステムである Survlog を開発した．Survlog は，各種の工夫を行うことで楽しく文献のサーベイができることや，各自の文献調査結果が整理された形で蓄積されていくことで関連する知識が自然に体系化される，というメリットを利用者にもたらすことを目指している．また Survlog は，授業の一環として実際のソフトウェアの開発工程に近い形でのプログラム開発を行うことで実践的に技術を身につけることを目的として作成されたものでもある．本論文では，我々がどのようなシステムを作成したのかについて述べ，どのような経緯や手段でこのプロジェクトが遂行されたのかについても述べる．

Survlog: a blog system for literature surveys

Hiromichi Goto,[†1] Kenta Hattori[†1]
and Masanori Matsumoto[†1]

1. はじめに

通常，研究者は研究活動を行うにあたって，過去に出版された多くの文献を調べ，知識の体系化を行う．体系化した知識を用いてまだ解決されていない問題や新しい問題を見つけ，それについて研究を行うのである．このような文献調査活動を行うにあたって，文献自体をどのように保存するかといった問題や，どの文献を読んだのか，まだ読んでいないのかの管理，読んだ文献の内容や感想のメモ，文献の分類分けなどの作業が必要となるが，これらをどのように行うべきかという方針は研究者によってさまざまである．

文献管理の例として，学会や著者のサイトからダウンロードしてきた PDF 等の電子ファイルを，内容によって分類された各ディレクトリに保存することで文献を管理し，さらに，内容の要約や感想などのメモをテキストファイルとして作成し，同じディレクトリに保存しておく方法，あるいは，市販の文献管理ツールを使用して管理するといった方法もある．しかし，これらの方法は，複数人数で同じデータを共有したり，感想を交換しあったりといったことはあまり想定されていないことが多い．もし，複数人数で情報の共有ができると，他の人の論文への評価やコメントも参照することで意見の交換が行え，次に読むべき論文を決定するときに参考にできると考えられる．また，このように情報を交換し合うことによって一人で文献調査を行うよりもモチベーションを高めることができると考えられる．他にも，前述の方法では論文の分類があらかじめ固定されてしまい，研究者独自の視点からの分類による整理が困難であったり，大ざっぱな分類しかできなかったりと柔軟性も低いといった問題点がある．各研究者の考え方にあわせた文献の整理や体系化のためには，より柔軟性を持ったシステムが望ましい．このような問題を解決するために，我々は柔軟な分類分けによって簡単に文献管理を行なうことができ，さらに，研究者同士の情報の交換や共有を支援するための文献サーベイブログシステム Survlog を作成した．

Survlog は，孤独な活動になることが多い文献調査活動を少しでも楽しくできるようにすることを第一の目標として開発を行なった．また，文献調査の要点は，幅広くあたった文献に対して，それぞれの研究者独自の視点をもって整理・消化し，体系化された知識を蓄積することにあると捉え，そのような活動を支援するようなシステムを目指して開発を行なった．

本論文では，我々がどのようなシステムを作成したのかについて述べ，また本システムがどのような経緯でどのようにして作成されたかについても述べる．なお，Survlog は現在学内のサーバに設置されており，http://survlog.org/ にて公開されている．

†1 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

図 1 Survlog 表紙

2. システム概要

我々が今回作成した Survlog は，論文やコメントなどを格納したデータベースとそれを表示，加工する Web アプリケーションとで構成されている．Web サイトとして構築されているので，ユーザはすでにインストールされている Web ブラウザのみで使用することができる．ユーザは各自でアカウントを作成することでシステムの持つ全ての機能を使用することができる．アカウントを作成しない場合でも公開されているデータを参照することは可能なので概要を見ることができる．Survlog の表紙画面を **図 1** に示す．また，ログインしたときに文献一覧がどのように表示されるかを**図 2** に示す．Survlog 自体を一つの文献と見立てたデザインが行われている．

図 2 文献一覧画面

2.1 システムの持つ機能

本節では文献の分類分けの柔軟性を向上させ，他のユーザと情報を共有させるために， Survlog にどのような機能を持たせたのかについて述べる．

2.1.1 文献の分類分け

我々は，文献の分類分けの柔軟性を向上させるためには，文献を大分類，小分類というように分類分けすることができるよう分類には階層構造が必要であると考えた．また，個々人が独自に分類分けが行えるということも重要だと考えた．

このような機能を実現するために，文献にタグをつけることができるようにし，タグは階層構造を持つことができるようにした．タグとは文献につける分類名のようなものであり，一つの文献は複数のタグを持つことができる．例えば，C++ コンパイラに関する論文の場合，「C++」「コンパイラ」といったタグをつけることが想定される．他にも，特定の目的のためにタグをつけることも想定されている．例えば，ある研究室に入ったときに，研究を行うにあたって読むべき定番の論文のリストが代々受け継がれてきたとする．このとき，このような定番の論文全てに「○○研究室定番論文」といったタグをつけておくと，簡単に検索ができるようになるので新入生に容易に情報を提供することができる．論文の情報以外に，論文を読むにあたって知っておくべき知識などのメモや論文自体の電子ファイルなどを同時に提供することも可能であるので，非常に効率の良い学習補助ツールとしても使用できるといえる．

タグ機能について，通常のシステムでは全てがフラット，つまり同一レベルで扱われることが多いが，Survlog ではタグに階層構造を持たせることができるようにした．例えば，「コンパイラ」タグは「コンピュータ」タグの子タグとして登録することができる．このようにすることで，文献の整理や体系化に役に立つ．また，この機能は柔軟な検索機能を提供することにも繋がっている．上記の場合，「コンピュータ」タグで検索すると「コンパイラ」タグのついた論文も検索される．階層構造を持ったタグで文献の管理を行った例を **図 3** に示す．

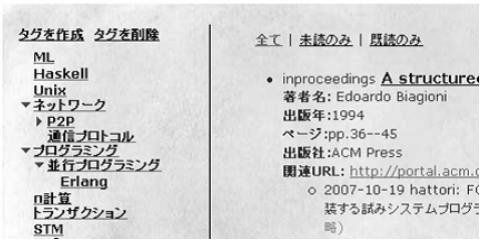

図 3 階層構造を持ったタグで文献を管理した例

さらに，ユーザはタグを公開タグにするか非公開タグにするかを選択することができるようにした．これにより，他人の分類分けを気にせずに文献の分類分けを個人的に行うこともできる．例えば，特定の論文を執筆する際に参考文献一覧を作成したいとする．そのとき，参考文献にしたい文献全てに同じタグ，例えば「prosym49」といった

タグをつけておけば後でそのタグで検索し，BibTeX 連携機能により BibTeX を出力することによって参考文献リストを容易に作成することができる．

2.1.2 情報の共有

Survlog では，文献を読んだときにその文献に関するメモや感想を登録することができる．メモや感想は文献と関連づけて登録されるので後でばらばらになることがなく簡単に整理できる．

我々は，ユーザ間の情報の共有を容易にするため，各人が文献に書いたメモや感想を全体に公開することができるようにした．これにより，同じ論文を後で読む人は先人のメモや感想を参考にして論文を読むこともでき，自分一人では気がつかなかった視点を得ることも可能となる．またこの欄を利用して議論を行うことも可能であり，より深い理解や新しいアイデアへと繋げることもできる．

他に，文献に星の数で評価を付けることにより，良い文献の情報を共有する機能も搭載した．評価が蓄積すると論文を読むかどうかを決めるための役に立つ情報となる．

2.1.3 その他の機能

Survlog に搭載した上記以外の機能について概要を述べる．

- 文献の既読管理

ユーザ毎に文献の未読・既読管理を行う．今までどれだけの文献を読んだのか，これからどれだけの文献を読むべきなのかが一目瞭然となる．未読の文献が蓄積することで心地よいプレッシャーを感じることができ，また既読の論文が日を追うごとに増えていく様を観察することで達成感を得ることもできる．

- ファイル添付

文献にファイルを関連付けることができる．PDF などの文献自体の電子ファイルや，文献を調べたときに作成したスライド資料などを添付することを想定した．また，ユーザはファイルを公開するか非公開にするかを選択することができるので，他の人に参照されたくないデータであっても添付することができる．

- BibTeX 連携機能

BibTeX から文献タイトルや著者，日付，出典といった文献情報をインポートして利用することができる．Survlog は BibTeX の各種エントリ種別を扱うことができ，多様な形式の文献が登録できる．もちろん BibTeX からのインポートだけでなく，手動で文献情報を入力することもできる．その場合は論文や書籍，報告書，マニュアルなどの種別を選択し，それに応じた項目を画面に従って入力する．また，指定した論文のリストを BibTeX 形式で出力することもできる．これにより，前述したように論文執筆時に参考文献一覧を作成することが非常に容易になる．

- 外部サイト連携機能

Survlog は現在 ACM Portal[⋆1] と CiteSeer[⋆2] から文献情報をインポートすることができる．それぞれのサイトで検索した文献の URL を入力することによりインポートを行う．

- リファレンス機能

文献をインポートしたとき，その文献のリファレンスも適切な形で管理したい．このため，文献の URL をキーとして，すでに同一の文献が外部サイトからインポートされている場合は自動的に Survlog 内でリンクが張られる．リファレンスは手動で編集することもできる．

3. システムの実装

Survlog はデータベースを利用した Web ベースのアプリケーション構築フレームワークである，Ruby on Rails[1] (以下 RoR) を用いて作成された．本節では RoR の概要について説明し，その上に実装した機能であるタグの階層構造での管理と外部サイトからの文献のインポートについての詳細を述べる．

3.1 Ruby on Rails フレームワーク

RoR のアプリケーションは Web サーバ上に配置され，RoR フレームワークから呼び出される．**図 4** で示されているように，ユーザは Web ブラウザを通して，Web アプリケーションの提供するサービスを利用する． RoR は

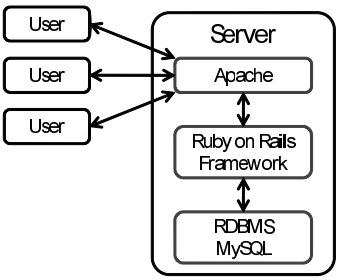

図 4 Ruby on Rails フレームワークの構成

少ないコード量で Web アプリケーションを作れるように設計されており，RoR を利用することで Web アプリケーションが比較的短期間で開発できると言われている．RoR フレームワークの上ではアプリケーションは Model View Controller アーキテクチャ[11] に基づいて構築する．また，OR マッピング[⋆3]により，データベースを直接操作することなくクラスの操作でデータベースの操作が自動的に行われるという利便性がある．

⋆1 ACM（国際計算機学会 : Association for Computing Machinery）が提供する文献データベース．
http://portal.acm.org/

⋆2 主にコンピュータサイエンスの論文が集積されている．
http://citeseer.ist.psu.edu/

⋆3 Object-Relational Mapping
オブジェクト指向のオブジェクトと関係データベースの関係モデルをマッピングする方法

3.1.1 Model View Controller アーキテクチャ

Model View Controller アーキテクチャでは，アプリケーションを Model, View, Controller に分離して別々に実装を行う．Model では保持するデータの構造[⋆1] を記述する．View ではデータをどのように表示するかを記述する．Controller ではユーザのアクション[⋆2]を受け取り，データを適切に加工するコードを記述する．

このようにアプリケーションを明確に分離することにより，これらの要素が密接に絡み合った従来の構造の設計手法と比べて遙かにプログラム全体の見通しが良くなるという利点がある．

3.1.2 RoR の採用

Survlog の開発にあたって，我々は，できるだけ開発工数を削減するため，特に生産性の高い Web アプリケーションフレームワークとして知られる Ruby on Rails を採用した．開発メンバーにとっては，最新の Web アプリケーション開発手法である RoR での開発のスキルを得るられるという利点もある．

RoR はその名前の示すように，プログラミング言語 Ruby[2] 上で動作するフレームワークである．Ruby はシンプルな文法を持つオブジェクト指向のスクリプト言語であり，他の言語に慣れているプログラマにとっては，習得は困難ではない．今回，プロジェクトメンバー 3 人のうち 2 人は，開発当初は Ruby の初心者であり，Ruby プログラムにも慣れていなかったが，比較的短時間でコードを書くことができるようになった．

RoR は「同じことを繰り返さない」(DRY:Don't Repeat Yourself) と「設定よりも規約」(CoC:Convention over Configuration) という基本理念を持っている．CoC では，標準的な設定は決まったルールに従って行われ，共通でない部分の設定のみ行うようにするべきと言うことである．この理念により，最小限の作業量で最大限の効果が得られる．例えば，決められた形式でデータベースのテーブルを作成して最小限の設定を行うことで，そのデータの編集画面や編集機能が自動的に作成される．また，データの定義を書き換えるだけでそれらも柔軟に変更される．

さらに，クラスも自動的に生成することができ，OR マッピングにより自動的にデータベースのテーブルが Ruby のクラス構造にマッピングされる．しかし，自動的に作成されるということは思わぬ副作用をもたらす場合がある．例えば，ユーザ毎に異なる機能を提供する場合，適切にアクセスコントロールを設定しなければユーザが権限を越えてデータを操作できてしまう画面が自動的に作成されるといった，セキュリティホールの要因となりうることが発生するので注意が必要である．自動的に作成されることは便利なことではあるが，初学者にとっては，このような暗黙のルールを理解して適切に使用するには，ある程度のコストがかかるといった問題点もある．

RoR では様々な人が作成したプラグインモジュールを利用することで簡単にパワフルな機能をアプリケーションに追加することができる．Survlog でもログイン処理やタグの階層構造を実現するために既存のモジュールを利用した．このようなモジュールはありのまま利用することは非常に簡単で利便性も高いものである．しかし，部分的に変更したい部分が発生しカスタマイズを行おうとするとモジュールのソースコードを理解しなければいけなかったり，モジュールの構造自体を変化させる必要も出てくる．

3.2 タグの階層構造での管理

残念なことに，RoR 上で階層構造を持ったタグを実現するための直接的なモジュールは，現在のところ提供されていない．このため，今回の開発では，より一般的な機能であるタグとツリー表示を実現するための RoR 上のモジュールを組み合わせて，階層タグを実現することにした．タグ機能を提供するプラグインではタグの情報を格納したデータベースのテーブルと，タグをどの文献に張ったのかを格納するテーブルを使用する．ツリー構造を実現するためのプラグインではツリーの親要素と要素の名前を格納したテーブルを使用する．今回はツリーの要素の名前はタグ名を使用したいので要素の名前の代わりにタグの ID を使用してテーブルを作成したい．しかし，これらのプラグインはそのような使用方法が想定されていなかったのでプラグインのソースコードを読んで対応するように修正する必要に迫られた．このように，フレームワークを用いた開発では，フレームワークが想定している方法に沿って開発する場合には効率よく開発することができるが，想定されていない使い方をする場合には効率が大きく低下する．

3.3 文献のインポート

文献情報のインポートを行うにあたって，当初は OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting)[3] というプロトコルを用いてデータを取得することを試みた．試しに CiteSeer から OAI-PMH を用いて情報を引き出したところ，タイトル，著者，登録日などの情報は取得できたが，ジャーナル名などの情報が取得できないことが判明した．ジャーナル名などの情報はブラウザ上で論文を検索したときには表示される情報であるので OAI-PMH では限定された情報しかアクセスできないと言うことである．

文献情報のインポートを行うには可能な限り多くの情報を取得するべきであるので，以上の理由により OAI-PMH を使用することをあきらめることにした．代替手段として，CiteSeer が出力する HTML データをパースして BibTeX で利用できる情報を抽出することでインポートを行うことにした．

このように，できる限り多くの情報を得るためには文献情報サイト毎にパースを行うルールを作成する必要がある．対応するサイトを追加する場合に備え，サイト専用のプラグインを作成して組み込める仕組みを構築した．

⋆1 データベースの構造であることが多い．

⋆2 Web アプリケーションでは Web ブラウザでの操作となる．

これにより，将来的に対応サイトを増加させることが可能となる．

4. ソフトウェア開発プロジェクト実践

Survlog は，情報理工実践プログラム[⋆1] の中のソフトウェア開発プロジェクト実践という演習コースの一つのテーマとして開発された．このプログラムは大学の授業の一環として，実際のソフトウェアの開発工程に近い形でのプログラム開発を行うことで，より実践的な技術を身につけることを目的としたものである．ソフトウェア開発のテーマは教員や学生からの提案にもとづいて実施される．本節では，演習としてのソフトウェア開発プロジェクトという視点から，Survlog の開発がどのようにして行なわれたか，また，プロジェクトを通じて得られた経験について述べる．

4.1 開発スケジュール

Survlog の開発プロジェクトは，2007 年 2 月〜3 月にかけての第 I 期，および，2007 年 5 月〜10 月までの第 II 期にまたがる，計 9ヶ月の期間で遂行された．第 I 期と第 II 期はそれぞれ大学の冬学期，夏学期に相当する．第 I 期の期間が短かったのは，情報理工実践プログラム自体の立ち上げ期であったという事情による．実践プロジェクトでは，中間報告会や成果報告会といったイベントが組み込まれており，学生らはこれらのイベントを 1 つのマイルストーンとして開発に取り組むことになる．

第 I 期は，開発期間が非常に短かったため，文献情報の登録や閲覧，コメントの記述といったごく基本的な機能の実装に的を絞って開発を進めた．いわば，プロトタイプとしての位置づけである[⋆2]．第 II 期の開発では，第 I 期のプロトタイプ版をベースにしつつ，外部サイトとの連携や，ファイル添付などの新たな機能の追加，より使い勝手をよくするためのインタフェースの改善などを行なった．

4.2 開発プロセス

Survlog の開発プロジェクトはあくまでも演習の一環であり，フルタイムのプログラマから構成される開発チームとは事情が異なる．我々の開発メンバーは学生（あるいは社会人学生）であり，それぞれ，講義やレポート課題，各自の研究など[⋆3]といったことに大きく時間をとられてしまう．その中で開発時間をやりくりし，開発をすすめなくてはならない．また，大学のプログラミング演習の課題や研究活動などにみられる個人的なプログラム作成とも異なり，ある程度の規模のソフトウェアを複数人で協力して開発していく必要がある．複数人での開発においてはメンバー間の意思疎通やミーティングなど，情報，意識共有のための時間が不可欠であるが，前述のとおり，各メンバーの抱える事情は様々であるため，ミーティングの時間の調整は一苦労である．

我々は，このような困難を克服できるようにソフトウェア開発を進めていかなくてはならなかった．一般的にソフトウェア開発のプロセスモデルとしては，伝統的なウォータフォールモデル[12] から，最近注目されている XP[9] やアジャイル[10] といった軽量の開発モデルが知られている．特に，アジャイル開発では，意味のない多くの文書を書くことよりも，プロジェクトメンバーが必要な時に即座に直接顔を合わせて意思疎通を行うべきであることを強調している．我々は「文書よりもメンバーの意思疎通が大切である」という考え方にもとづいて，以降で述べるように工夫して開発を進めた．

4.2.1 ミーティングの工夫

前述のように，メンバーは各自の予定を抱えているため，メンバー全員が集まってのミーティングは，お互いにフェースツーフェースのコミュニケーションが図れる非常に貴重な時間である[⋆4]．したがって，この機会を無駄にすることはできない．

ミーティングはホワイトボードとプロジェクタを積極的に活用することによって，設計の議論を深め，メンバー間で設計内容や方向性を共有できるようにした．仕様検討や設計の議論においては，逐一，意見を出し合いながら，ホワイトボードにその内容をまとめていった．**図 5** は，そのようなミーティングの一コマである．

図 5 ホワイトボードを前にした議論の風景

また，**図 6** に機能項目を決定する際のホワイトボードがどのように使用されたかの例を示す．

ホワイトボードに書かれた機能項目や設計などの内容は，その都度デジタルカメラで記録しておき，qwik（後述）のミーティング記録用のページに添付し貼り付けておいた．この記録は内部の設計ドキュメントの役割を果たすものである．我々のやり方は，ある担当者が設計ドキュメントを作成し，それを他のメンバーでレビューすると

⋆1 http://www.i.u-tokyo.ac.jp/ist_hands-on/

⋆2 ちなみに，第 I 期のメンバーは服部，松本の 2 名であり，第 II 期になって五嶋が加わった．

⋆3 あるいはバイトなど:-)

⋆4 多くても週に 2 回程度の頻度でしか実施でなかった．

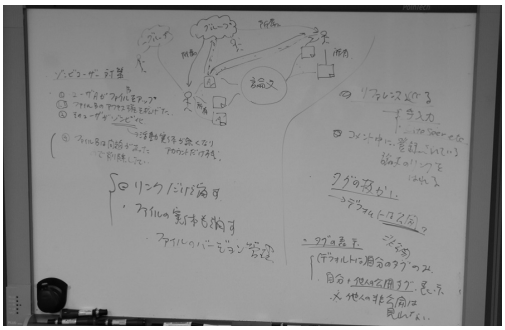

図 6 ホワイトボードへの書き込み例

いった一般的な方式にくらべて，その手間を大幅に削減する．さらに全員で設計の議論を行なったために，その内容は自然にレビューされるとともにメンバー全員で共有されるという利点がる．このようにして，我々はほとんどの内部ドキュメントを作成する手間を省くことができた．ただし，「無駄なドキュメントは作るべきではないが，必要な最低限のドキュメントは作るべきである」という方針にしたがい，DB のスキーマ設計など，重要な情報については，ホワイトボードの写真を添付するだけではなく，その情報を qwik 上に整理することでより参照しやすい形で記録を残した．

4.2.2 進め方の工夫

プログラムを書き始める前に，どのようなものを作成するべきかといった青写真を描き，メンバー全員で共有するに念入りにミーティングを行い，実装すべき機能項目の一覧を作成した．機能項目には重要度を設定し，重要度の高いもの順に実装を行うようにした．作成するべきものがある程度固まった後では，毎週のミーティングで進捗状況を報告し，次週までにどのような作業を行うかを決め，タスクの割り当てを行った．

もちろん実装の途中で新しいアイデアが出てきたり，実装の重要度が実は高くないと判断し直す場合もあるので，その場合は柔軟に実装すべき機能項目の変更を行った．どのような機能項目一覧を作成したかについて**図 7** に一部分を示す．

4.2.3 使用ツールの工夫

プロジェクトは主に qwik[4] を用いて管理した．qwik は Wiki[⋆1] とメーリングリストを融合したものであり，メーリングリストのログと，開発に使用した各種ドキュメントが同時に管理できる利便性があり非常に有用である．qwik の上に開発計画書やデータベースモデル，機能仕様書，ミーティングの議事録，月報，作業記録などすべてのドキュメントを格納した．これによって，プロジェクトのメンバーはいつでも必要なドキュメントを手軽に閲覧することが可能になる．より本格的なプロジェクト管理ツールとしては Trac[5] があり，本プロジェクトでも開発用サイトを Trac で構築していたが，結局あまり使われず，より軽量な qwik に収斂した．

複数人による開発では，ソースコードの共有とバージョン管理が重要になる．ソースコードの管理には，最近，広く使われるようになった Subversion を用いた．これによって，各自が好きなタイミングで自由にソースコードを編集できるようになった．また，ソースコードのコミット時には，各自が変更箇所をコメントとして残すことで，後から変更履歴を容易に追跡できるようした．

4.2.4 作業分担の工夫

複数人数で作業を行うメリットとしては，やはり作業を分担して一人あたりの負荷を減らせることである．その代償として，一人での開発の場合は必要のない，各自の目標の設定，タスクの割り当てといった作業が必要となる．

今回はやるべき作業や実装をプロジェクトの初期段階でリストアップして分離しておき，そのリストの項目ごとに担当者を割り振ることで作業の分担を行った．近い機能の実装はなるべく同じ人が行うようにすることや，各人のスキルに応じて比較的得意な作業を割り当てることにより効率よく作業を分担することができた．

作業分担の問題として，大きい粒度でタスクを担当者に割り当ててしまうと，担当者間の進捗状況にばらつきが生じた場合に，やることが無くなってしまう人や，逆に，難しいタスクを割当てられて貧乏くじを引いた人には負荷が集中してしまうといった問題が発生しうる．これを回避するためには，タスクをなるべく細かい粒度まで分割して，割当てる方法が有効である．基本的に設計内容をメンバー全員が共有し，理解しているので，非常に柔軟に担当者の割り当てを行なうことができた．

4.3 開 発 合 宿

第 II 期開発の後半では，千葉県上総一ノ宮にて一泊二日の開発合宿[⋆2]を行い，成果報告会に向けて，集中的に機能を実装し開発を加速させた．合宿では，会議室を一部屋借りた上で，そこにルータや PC などの機材を持ち込み開発環境を整えた．合宿の集合時に，まずはメンバーとともに合宿における開発目標を決め，その達成に向けて各自作業を行なった．**図 8** は，合宿での開発風景である． チームのメンバー全員が一箇所に集まって作業を行なうため，技術的，仕様的な問題が発生した場合には，すぐに相談することができ，その結果，迅速に開発を進めることができた．普段の開発では，メンバーが一堂に会して実装を行なう機会というのはほとんど無かったため，この合宿は非常に貴重なものとなった．

4.4 デザインの外注

Web アプリケーションにおいて，デザインのよしあしは，多くのユーザに受け入れられるかどうかの鍵を握っており，重要な要素のひとつである．残念なことに多く

⋆1 Web ブラウザ上で複数人が共同して Web サイトを構築するためのシステム．

⋆2 この開発合宿は他のプロジェクトチームと合同で行なわれた．

重要度の定義(難易度も織り込み済み)

- *** -- 必ず実装する(must)　** -- できれば実装する(recommended)　* -- 余裕があれば実装する(optional)

	分類	機能項目	説明	重要度	担当	進捗
1	リファレンス機能					
1.1		リファレンス入力	文献のリファレンスを入力する	***	服部	済
1.2		リファレンス表示	文献のリファレンス一覧を表示する	***	服部	済
1.3		リファレンスリンク	リンクをクリックすると参照先にとぶ	***	服部	済
1.4		文献DBとの連携	CiteSeerなどのサイトから連携し，自動入力	***	服部	済
2	検索機能					
2.1		論文の全文検索	Paperテーブルの全フィールドから検索する	**		
2.2		タグ検索	タグ付けられた論文を検索する	***	服部	済
2.3		コメント内の検索	コメントを全文検索する(要検討)	*		
2.4		条件検索	AND,OR,NOTで条件指定して検索できる	**		
2.5		検索結果ソート	検索した結果を評価値の順番などでソートする	**		

図 7　機能項目一覧

図 8　合宿の風景

のプログラマはデザインセンスに欠けており，商用ソフトウェアのような専門のデザイナを雇うことができない，オープンソースソフトウェアでは，そのデザインがどうしても貧相なものになってしまう．Survlog プロジェクトでは，プロジェクト実践の予算によって，デザインを専門家に外注することができた[*1]．今回は，産業技術総合研究所 櫻井 稔氏と東京藝術大学 杉山 聡志氏に，デザインチームとして参加して頂だき，Survlog のロゴやページデザインを作成していただいた．

デザインを決めるに当たってはまず，デザインチームが我々開発メンバーの持つ漠然としたイメージを引き出すためのインタビューが行なわれた．たとえば，何を作ったのか，ターゲットは誰か，システムの売りは何か，どう見られたいのか，開発者各人が持っている Survlog に対するイメージなど，こと細かな点を聞き出していった．インタビューに一つ一つ時間をかけて答えていくうちに，たとえば，イメージでは，「人間が自分の頭を使い，汗を流し，読んで理解し，記録を残していくそのための道具」「手になじむもの」「金属よりも皮等のぼろぼろな手帳」「使い込むと味が出る」など，開発中には，つきつめて考えたことがなかったシステムイメージに関して，メンバー同士の意識あわせが行われた．

そして，これらのイメージをもとに，ファイリングキャビネット，積み重ねられた本・論文，古文書などのイメージを元にした，いくつかのキービジュアル[*2] 案が作成された．その中からさらに，我々とデザインチームとの間でブレインストーミングを繰り返しながら候補を絞っていき，最終的なキービジュアルが決定した．デザインチームは，その後，画面イメージの作成，CSS 作成と画面の調整を経て動作確認，メインブランチにマージ，マスターアップとなった．

今回は，デザインの外注が決定したのが 8 月末だったため，成果報告会まで，短い時間しか残されていなかった．したがって，デザインチームとのミーティングの回数を十分に確保できず，見切り発車的に進んでしまった面もある．さらに，こちらの考えているデザインのイメージを言葉で相手に伝えるのは難しく，特に最初のインタビューでは，我々のイメージがデザインチームに伝わったかどうか，非常に不安であった．しかし，具体的なキービジュアルが提示されると，それによって我々のイメージもふくらみ，活発な議論を行なうことができた．今回は時間が非常に限られていたので苦労したが 初めての経験であり，また，プログラマにはないデザイナー独自の考え方や感性に触れることができたのは良い経験となった．

4.5　プロジェクトの総括

実践プロジェクトのような，複数人による開発のメリットとしては，システムの設計を行うに当たり，全員でアイデアや意見を出すことで多様な考えが出て多面的な視点をもって機能を取捨選択することができたことである．これは一人での開発では難しいところであった．逆に，このような議論で多くの時間を費やしたことで，実装に割ける時間が削られてしまい，いくつかの機能の実装が間に合わなくなってしまったという面もある．これは反省点となった．

⋆1 この僥倖は，本格的なプロジェクト開発においては，外注管理も重要であるという教育的配慮からきている．

⋆2 鍵となる図柄や画像のこと．今回の場合トップページ (図 1) に該当する．

今回はたかだか三人ではあったが，実際に複数のメンバーで開発を行うと一人の場合とは勝手が違い，コミュニケーションコストが発生する．特に，各メンバーの持つプロジェクトのゴールイメージが異なってしまうと，そのコストはより大きいものになるだろう．Survlog の開発ミーティングでは，各メンバーの持つ，チームとしての方向性を常に共有しておくように気をつけて行うようにした．特に開発の前半では，Survlog の目指す方向性の議論に多くの時間を費やしたが，そのおかげでメンバーの意識が共有され，後半では，そのような議論をほとんど要することなく，スムーズに実装を進めていくことが可能になった．

5. 関連ソフトウェア

近年，学術的文献情報のデータベース化が進み特定の主題に関する文献を一度に網羅的に検索することが可能となってきており，この検索結果のリストを活用することを目的としてさまざまな文献管理ソフトウェアが開発されている．[6] 市販のものでは Endnote[7] が代表的であるが，これは学術文献の情報検索と検索結果のダウンロードファイル管理を行うほか，たとえば Microsoft Word と連携し論文作成時の引用文献欄と参考文献リストを自動作成するといった機能ももつ．しかしながら各人のデータを積極的に共有するものではなく，タグ付けやファイルのアップロード機能などの機能も存在しない．

一方，文献情報に関する知識共有に主眼をおいたものとしては，安田絹子氏による論文サーベイ INDEX[8] というサイトがあげられる．これは，「論文や技術文書 (コンピュータ系で英語のもの) を読んだとき， そのサーベイ資料・解説・感想などのコメントを登録していって知識共有をはかる」ことを目的とし，文献情報およびそのコメントの登録・閲覧を行うためのサイトである．取り扱う文献はコンピュータ系限定であり論文の分類は固定的で階層構造をもたせることはできず，またユーザが自由に個人個人の既読・未読情報管理を行うこともできないなど文献情報整理をサポートするための機能としては限定的である．我々はこのサイトを意識しながら，より柔軟に，楽しみながら論文を整理するために Survlog を開発した．

6. ま と め

本研究では，文献調査活動を少しでも楽しくできるようにすることを目標として文献調査活動をサポートするシステムである Survlog を作成した[⋆1]．研究者の考え方にあわせた文献の整理や体系化を容易に行うために階層的なタグ付けという機能を提供し，また研究者同士の情報交換や共有を支援するために文献に対するコメントを全体に公開することや文献の評価を手軽に行えるようにした．

現状では個人もしくは全体という二者択一の公開設定しかできないためにグループ内だけで共有したい情報を作成することは不可能であり，これは今後の課題となっている．この機能が実現されるとタグのツリー構造や文献ファイルなどを特定のグループ間で共有することができるようになり，特定のグループ間での情報の交換や共有をさらに進めることができると期待している．ただ，どのようにして実現させるかについてはデータベースの構造を含め，今後綿密に検討を行う必要がある．

Survlog の開発は授業の一環として行われ，開発に使うドキュメントはどのようなものをどのように作成すべきかということや，ミーティングをどのように行うか，プロジェクトをどのように進めるかについて考え，検討を重ねることができた．このようなことは普段の研究活動では得難い経験であり有意義な体験であったといえる．

謝辞 文献サーベイブログ Survlog は文部科学省「先導的 IT スペシャリスト育成推進プログラム」による「情報理工実践プログラム」の一環として研究開発された．また，Survlog の開発にあたっては，稲葉真理准教授と笹田耕一助教から，数多くの有益なアドバイスと励ましをいただきました．ここに感謝の意を表します．

参 考 文 献

1) http://www.rubyonrails.org/
2) http://www.ruby-lang.org/
3) http://www.openarchives.org/OAI/openarchivesprotocol.html
4) http://qwik.jp/
5) http://trac.edgewall.org/
6) http://www.library.tohoku.ac.jp/mylibrary/tutorial/2004/st-furoku02.pdf
7) http://www.endnote.com/
8) http://www.spa.is.uec.ac.jp/kinuko/survey/
9) Kent Beck and Martin Fowler. *Planning Extreme Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.
10) Alistair Cockburn. *Agile software development*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
11) Trygve Reenskaug. Thing-model-view-editor an example from a planningsystem. 1979. http://heim.ifi.uio.no/~trygver/1979/mvc-1/1979-05-MVC.pdf
12) W.W. Royce. Managing the development of large software systems: concepts and techniques. In *ICSE '87: Proceedings of the 9th international conference on Software Engineering*, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

⋆1 開発者の一人である服部も博士論文の執筆に使用している．