Making of CS2

試合結果への熱き思い

総合政策学部 4 年　阿部　雅美 (Member)

環境情報学部 3 年　有田　直弘 (Member)

政策・メディア研究科 修士課程 1 年　杉浦　学 (Project Manager)
webappli2003fall-cs2@crew.sfc.keio.ac.jp
http://etude.crew.sfc.keio.ac.jp/cs2/
CreW Project

2005 年 4 月 7 日

目次

文責リスト

各章の分担

1 章　分析　有田
2 章　設計　阿部
3 章　実装　有田
4 章　評価　有田
5 章　感想　阿部　有田

コラムの分担

HCP チャートによる設計（2 章　設計）　有田
サッカーに関する英語について（3 章　実装）　阿部
CSV ファイルを読み込むオートマトン（3 章　実装）　阿部

第 1 章

分析

1.1　プロジェクトの開始

1.1.1　背景

CreW（大岩研究室）にはプレーステーション 2 用のサッカーゲームが置いてあり、それをプレーする人は多い。長い間プレーしてきて、誰が最も強いのか、自分は CreW の中でどれくらいの強さなのか等はっきりさせたいという声が CreW 内からあがり、勝敗を記録するためのシステムを作ってはどうかという提案が CreW 側からなされた。

1.1.2　企画

企画はシステムの目的を明確にする作業から始まった。やはりユーザーを幸せにすることを第一に考えると、誰が強いかという揉め事をなくし、プレイヤー全員を仲良くすることを目的にするのが良いのではないかという意見がプロジェクト内で出た。しかし研究室中の進捗報告にてこの旨を伝えると、仲良くしようとは考えなくていいとの意見を数人から頂いた。そこでこのシステムの目的は勝ち負けをはっきりさせ、誰が強いのか一目瞭然にすることに決めた。勝ち負けをはっきりさせるだけで、それによって仲良くなるとは保証しないし、揉め事が起こっても我々は関知しないことにした。

まずプロジェクト名を決めた。候補は 30 以上あげられ、「みるみる」のような柔らかい感じのする名前が多かった。以下がその候補である。

- 俺俺（オレオレ）
- ロスタイム
- オフサイド
- ハーフタイム
- 記録したら仕事
- 記録しろしろ
- きろーく君

- きろくん
- キロクン
- キロキロ
- キロリン
- きろりん
- れこーどきろりん
- きろきろ
- ジーコ君
- サントス君
- サッカー君
- 記録したら働け（きろはた）（きろけ）
- 記録後に働け
- スコアみるみる
- 戦績
- CreWSoccer
- 鬼のいぬま
- scorer
- scorekeeper
- すコアラー
- 喧嘩はやめて
- キロップ

これだけの候補があがってもメンバー全員が納得する名前は出なかった。そこで「みるみる」のような名前は諦め、これらの候補の中のものを合わせて「CS2 -CreW Soccer Scorer-」という無難な名前に決定した。

次に企画書を作成した。阿部と有田、二人とも企画書案を書き、プレゼンテーションを行い、レビューを行った。そして良いところを合わせて企画書を書き上げた。 完成した企画書を図 1.1、図 1.2、 図 1.3 に示す。

企画書

2003/10/08

Project "CS_2"

PM : Manabu Sugiura, MEMBER : Masami Abe, Naohiro Arita

システム名

CS_2 - CreW Soccer Scorer -

概要

CreW の研究室にあるサッカーゲームをする人が、そのゲームの対戦結果を簡単に入力でき、その情報からランキングを作成し表示するシステム。ランキングを見ることでゲームをする人は CreW の中で自分がどれくらいの実力なのかを知ることができる。

目的

CreW の研究室でサッカーゲームをする人たちの、ゲームの上手さ、力関係をはっきりさせること。

ターゲット

CreW の研究室にあるサッカーゲームをする人

背景

CreW に所属している多くの方々が仕事の合間に研究室でサッカーゲームをしている。長期間やってきて、誰が一番強く、そして自分はどれくらい勝っているのかがわからなくなってしまっている。しかし試合の結果を紙に記録し、勝率などを計算することは手間がかかってしまうので誰もやらない。

機能

「必須な機能」

○勝敗の記録をつける

- 試合の結果を入力する

○記録を見る

- プレイヤーごとの通算成績を表示する
- 全プレイヤーのランキングを表示する

図 1.1: 企画書

「あったら良いと思う機能」

○勝敗の記録をつける

・ モバイル端末での入力を可能にする

・ 得点、PKの有無、使用チームを入力できる

○記録を見る

・ １試合平均得点などの詳細なデータを表示する。

「それ以外の機能」

○勝敗の記録をつける

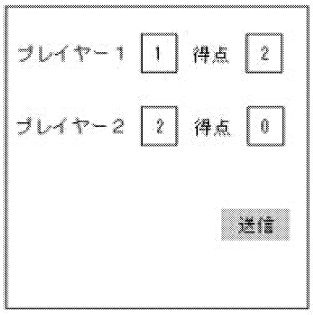
・ 前半後半、延長戦の有無を入力できる

○記録を見る

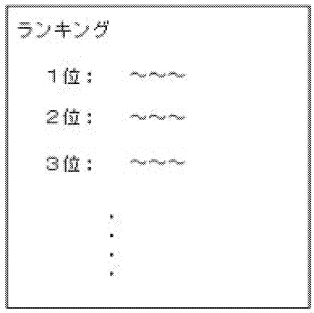
・ 月ごとのランキングを全プレイヤーにメール配信

画面イメージ（携帯端末用の画面）

入力画面



ランキング表示画面



計画

１０月中

企画書作成

設計作業

１１月中

実装作業

中間発表までに CUI 版の完成

その後、改良、Web アプリケーションへの移植作業

１２月中

テスト

プロトタイプ導入

図 1.2: 企画書

１月中

ドキュメンテーション

まず、ゲームをしている当事者たちへのインタビューを行う。次に記録に必要な要素を書き出した「記録表（紙バージョン）」を作成し、実際に使ってもらう。そして余計なものを省いていき、最低限の要素に絞った段階でアプリ製作に入る。何度もテストを繰り返し、完成品へと仕上げていく。

図 1.3: 企画書

また、システムのトップページ等あらゆる場所で必要になるロゴを作成した。ロゴはそのプロジェクトのイメージとなるので非常に大切である。そしてロゴを作ることでメンバーのモチベーションが上がるという効果もある。候補となる２つのロゴを作成し、選ばれた１つをより見やすく、すっきりとしたものになるように何度か修正し完成させた。(図 1.4)

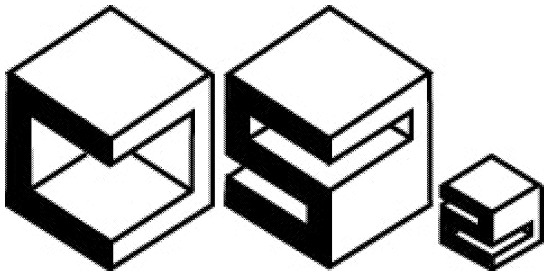


図 1.4: プロジェクトロゴ

1.2　仕様の分析と決定

1.2.1　要求把握

1.2.1.1　入力情報に対する要求

試合結果を提示することが中心となるシステムなので、結果についての入力情報は重要なものとなる。ユーザーがどのような情報ならば面倒臭がらずに入力するのかを把握し、そしてその情報からどのような種類の表示結果を出力できるかを考えるためにまず何も書かれていない紙に試合結果を記録してもらった。 それが図 1.5 である。

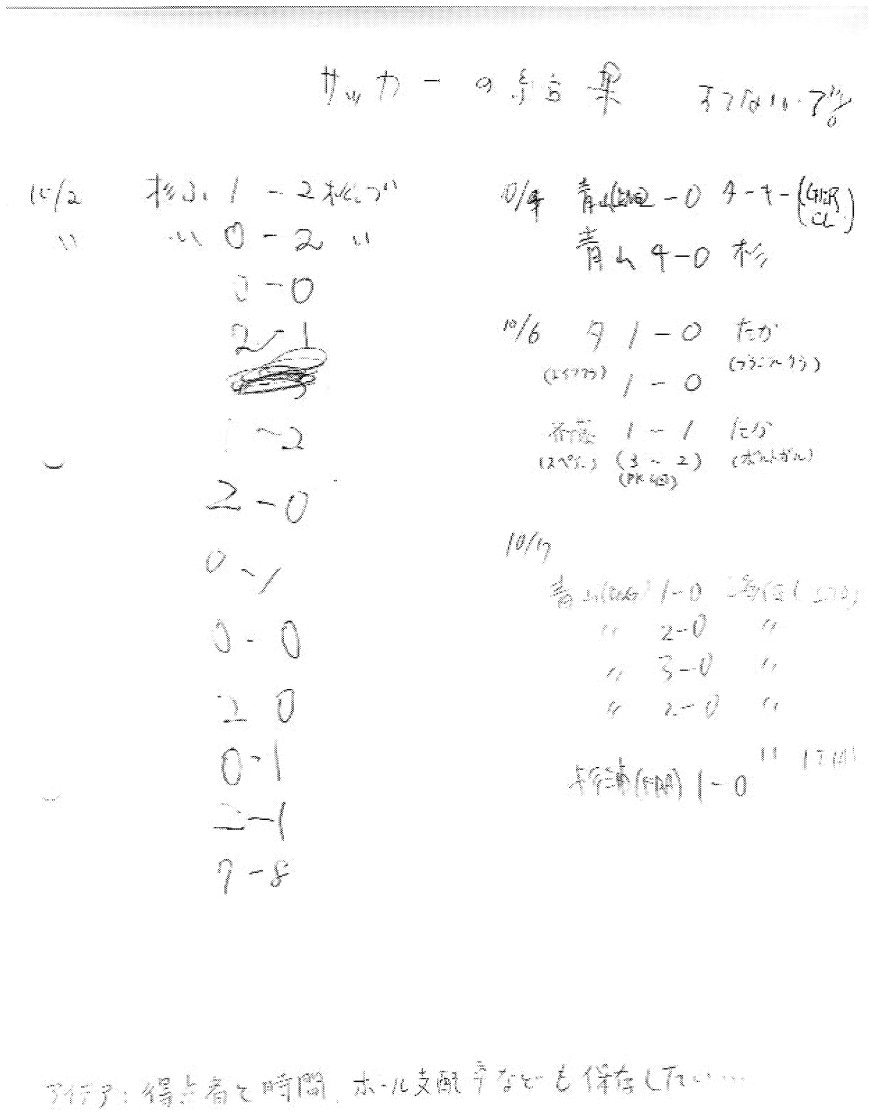


図 1.5: 手書きスコア

図 1.5 には「日付」「対戦者 2 名の名前」「得点」そしていくつかには「使用チーム」が書かれている。これらがユーザーの要求する入力情報だと判断し、手書き入力シートを作成した。また、「日付」「対戦者 2 名の名前」「得点」の情報からランキングが作ることができると考えた。 それが図 1.6 である。

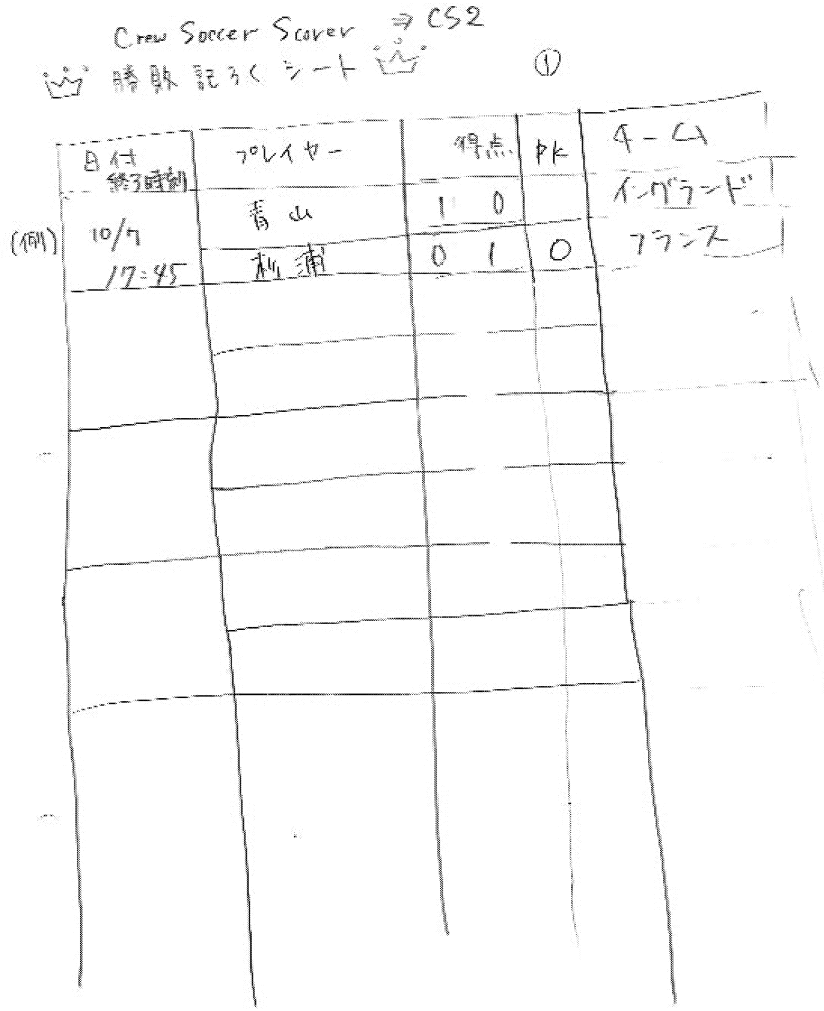


図 1.6: 手書き記入シート案

ミーティングで使用チームを入力する必要が本当にあるのか議論になり、使用チームは入力情報から除かれることが決まった。なぜならば、このシステムの大きな目的は誰が強いかをはっきりさせることで、プレイヤーがどのチームを使ったかは関係ないからである。この決定は企画の段階で目的を明確にした恩恵に他ならない。 そのため、手書き記入シートは図 1.7 のように変わった。

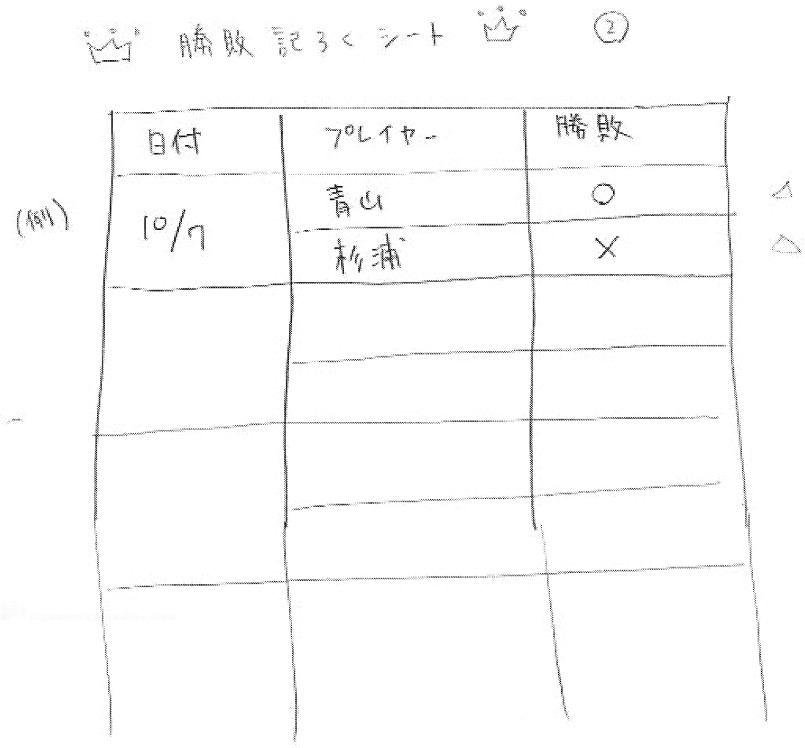


図 1.7: 手書き記入シート案改

1.2.1.2　ランキングに対する要求

ユーザーが手書き記入シートを使用して、ある程度試合結果が溜まった頃、とりあえず現在までのランキングが見たいから作って欲しいとユーザーに言われた。実装のためのシミュレーションとして自分たちの手でランキングを作ることは、ランキングを計算するためには具体的にどのような要素、数値が必要かを洗い出すことができたりと大切な点が多い。

どのようなランキングが必要かは話し合いで決め、「勝率」「総試合数」「勝利数」「敗北数」「総得点」「総失点」「平均得点」「平均失点」の項目を作ることとなった。実際に手集計でランキングを作ってみて発表したところ、他の種類のランキングが欲しいとの意見はユーザー側から出なかったので、ランキングは上記のものに決定した。

1.2.2 仕様の決定

1.2.2.1 入力方式の決定

提案を整理するために研究会中に使った提案書を図 1.8、 図 1.9、図 1.10 に示す。

ユーザーに入力してもらう情報はプレイヤー ID と得点である。これをどのように入力してもらうか議論がなされた。以下にその候補と長所・短所を示す.

コード入力 当初想定していた携帯電話からアクセスしての入力。数字だけ打ち込んでもらうだけでよいのでユーザビリティは高い。しかしながら、もしも使用チーム等、数字では表しにくい項目を入力することになった場合、非常に対応しづらくなってしまう。

選択式 プレイヤーと得点をプルダウン式にしてそこから選ぶようにする方法。これは一見使いやすいように思えるが、プレイヤーが増えた場合、縦に伸びて選ぶのが大変になる可能性がある。あまり有効なインターフェイスになるとは思えない。

バーコード入力 操作性が良いが、バーコードリーダーは他のプロジェクトが使用していて、さらに入力用のマシンを常に起動させていなければならないというデメリットがある。

タッチパネル式 おそらくユーザーにとってはこれが最も入力しやすいものだろう。だが、このプロジェクトのためだけに購入する必要があるとは思えない。

コード入力を採用しプロトタイプ版を作成すれば、GUI に移行した時にどの入力方式へも変更しやすいとの理由から、コード入力で作っていくことが決まった。

入力方法候補提案書

2003/10/08

Project "CS_2"

PM : Manabu Sugiura,　MEMBER : Masami Abe,　Naohiro Arita

1．コード入力

当初の案。携帯電話を想定。ユーザーに数字だけを入力させる。

プレイヤー１［1］得点［1］

プレイヤー２［4］得点［1］[送信]

※ユーザー登録時にユーザーの番号を決めておく

さらに簡略化して

［1 4 1 1］[送信]

というインターフェイスも考えられる。

しかしこれでは、10対0というスコアになった時、数字の区切り目を認識できなくなってしまう。これを解決するために次のような入力方法を提案する。

［1＊4＊1＊1］[送信]

ユーザーが米印を入力し、区切る。これにより二桁の数字入力も可能となる。

○メリット

・数字を立て続けに入力するだけという操作の簡単さ

○デメリット

・数字だけだと入力の項目が限定されてしまう

・簡略化バージョンだと入力する数字の意味、順番をユーザーが覚えなくてはならない

図 1.8: 入力候補提案書

2．選択式

誰と誰とが対戦し、そのスコアはいくつだったのか、という情報の種類は通常それほど多くない。一覧から選択できるようなインターフェイスにする。

[青山ー杉浦][青山ー海保]… それ以外［ ］ー［ ］[送信]
[０－０][１－０][０－１]… それ以外［ ］ー［ ］[送信]

○メリット

・候補にあげられているスコアになれば、ユーザーにとってわかりやすいインターフェイスとなる

○デメリット

・ユーザーが増えると、一覧に載る候補が増えてしまう。
・「それ以外」のデータを入力するのが面倒になる。

3．バーコード

携帯電話を使わずに、すべての情報の入力をユーザーがバーコードシートから読み取ることで行う。

○メリット

・操作がとても簡単
・入力の項目を増やしてもユーザーにとって苦にならないと思われる
・大岩先生の Libreto を提供していただければ省スペースに収まる

○デメリット

・バーコードは他のチームも使う（新たに購入する必要が？）
・バーコードシートがすぐ劣化する可能性がある（汚れ、破れ）
・記録用のマシンを常に起動させていなくてはならない

4．タッチパネル

ATM などで見かけるタッチパネルを使い、ユーザーは画面を押すだけですべての情報の入力を行うことができるようにする。

図 1.9: 入力候補提案書

○メリット
・非常にわかりやすいインターフェイス
・入力の項目を増やしてもユーザーにとって苦にならないと思われる

○デメリット
・現在、研究室にはタッチパネルがない
・このプロジェクトのために購入するのはいかがなものか。

図 1.10: 入力候補提案書

1.2.2.2　画面遷移の決定

入力方法が決まったところで画面遷移図を考えた。まず PM 杉浦を含めた 3 人で、自分が想像する画面遷移図を紙に書いた。 それが図 1.11、図 1.12、図 1.13 である。

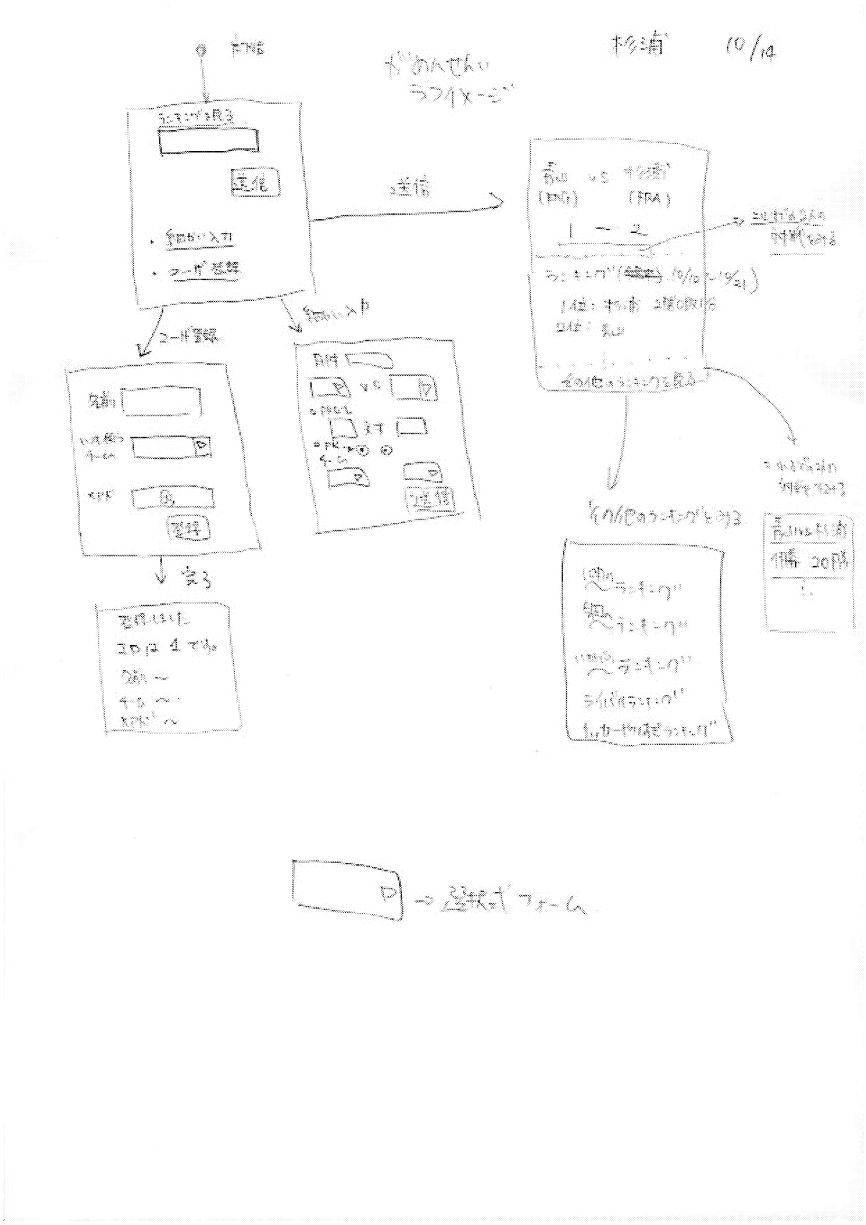


図 1.11: 手書き画面遷移図案（杉浦）

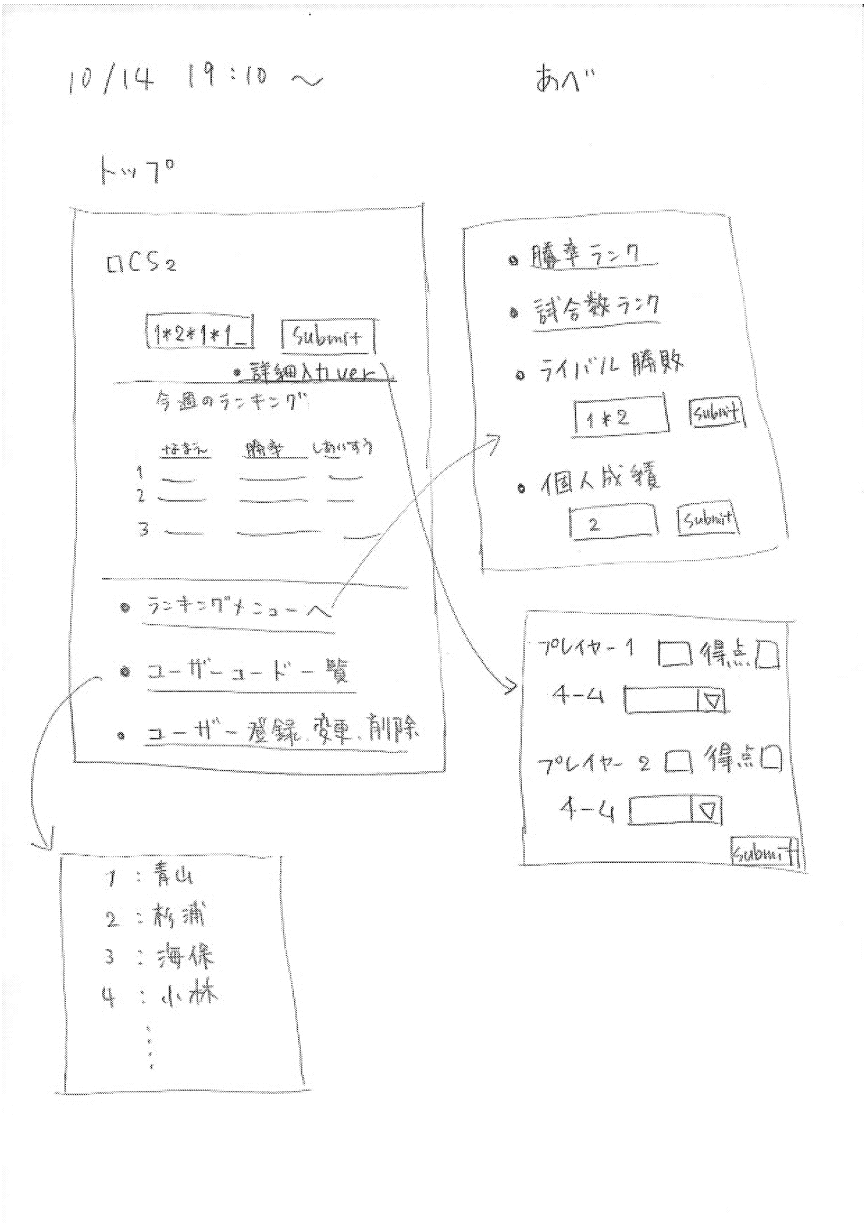


図 1.12: 手書き画面遷移図案（阿部）

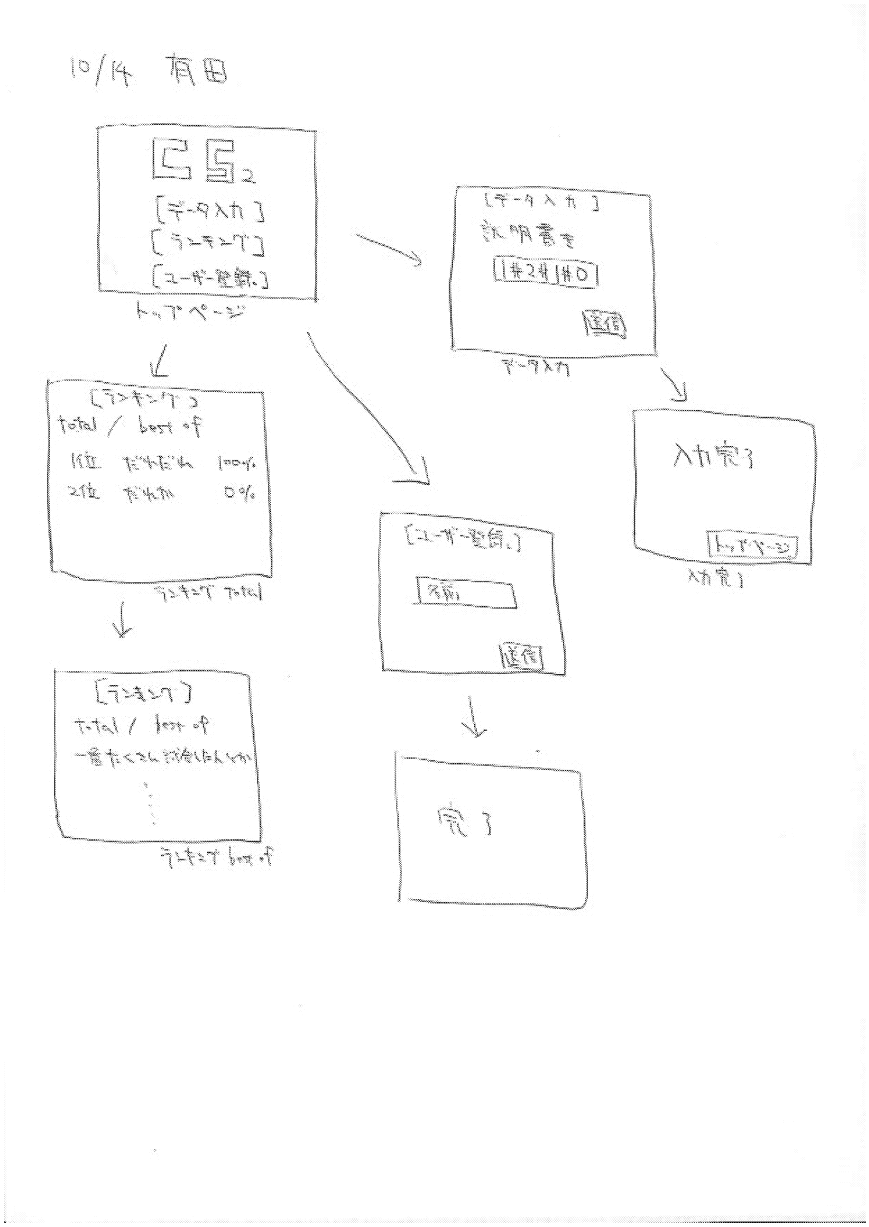


図 1.13: 手書き画面遷移図案（有田）

自分の書いたものをプレゼンテーションし、 他の 2 人はレビューを行い、1 つにまとめた。ここでランキングはトータルと最近一週間と最近 1 ヶ月に分けることに決まった。それが図 1.14、図 1.15、図 1.16 である。

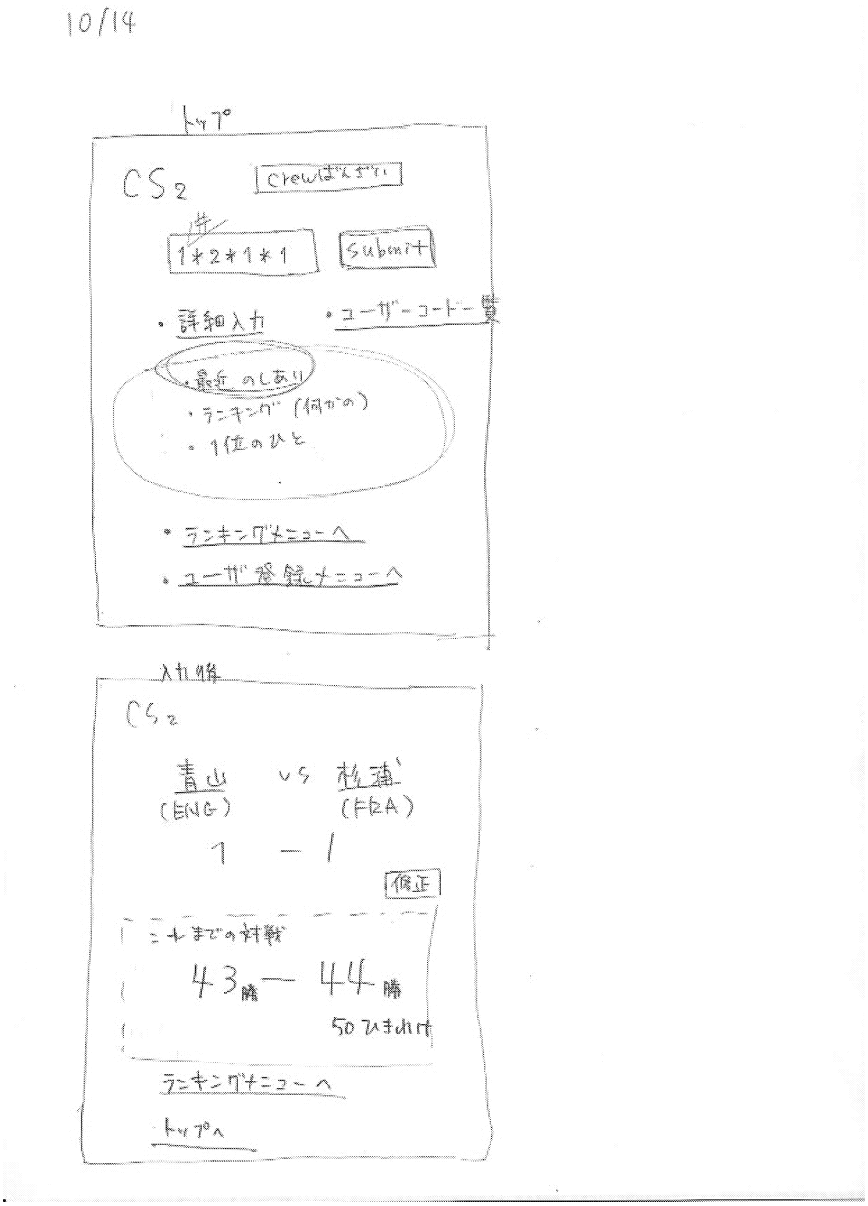


図 1.14: 手書き画面遷移図統合版 1 （阿部）

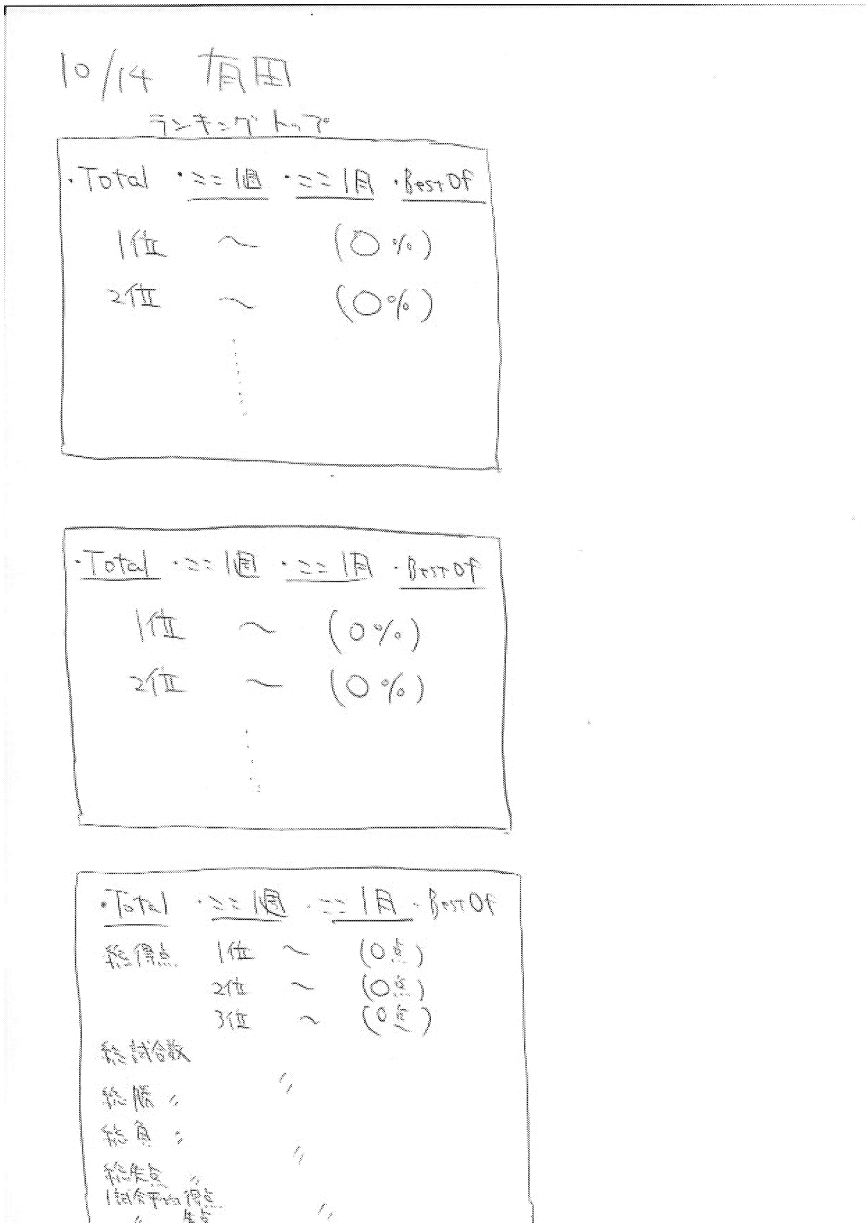


図 1.15: 手書き画面遷移図統合版２（有田）

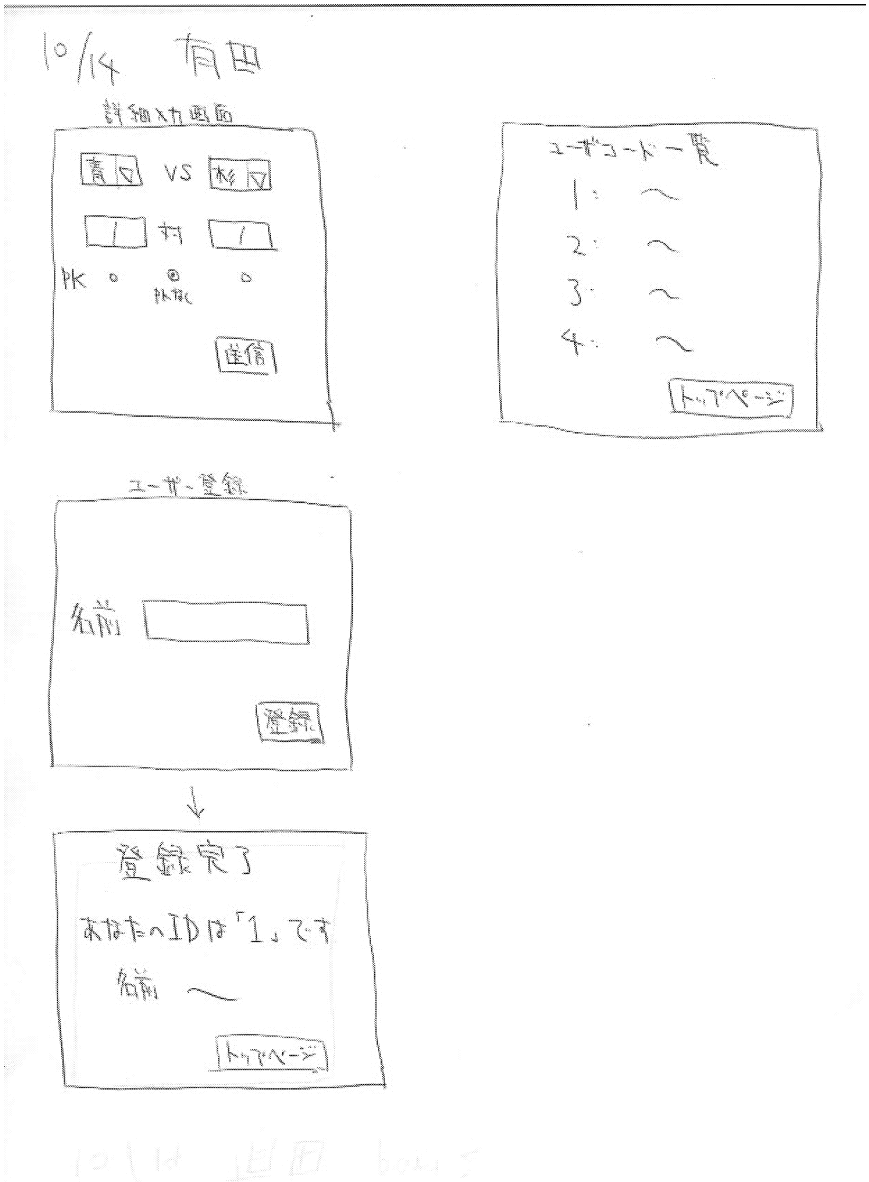


図 1.16: 手書き画面遷移図統合版３（有田）

上記の画面遷移図は手書きで、リンクとなっている場所やどこにプルダウンがあるかわからないので、それらがはっきりとわかるように清書した。（図 1.17、図 1.18、図 1.19 ）

プロトタイピングが終わった頃、パーソナルページ化することが決まったので、画面遷移図は大きく変化した。パーソナルページ化とは登録されている各ユーザーがそれぞれのトップページを持ち、そのユーザーの情報を見やすくするというものである。パーソナルページトップをブックマークしてもらうことを前提として画面遷移図を作成した。（図 1.20、図 1.21、図 1.22、図 1.23）

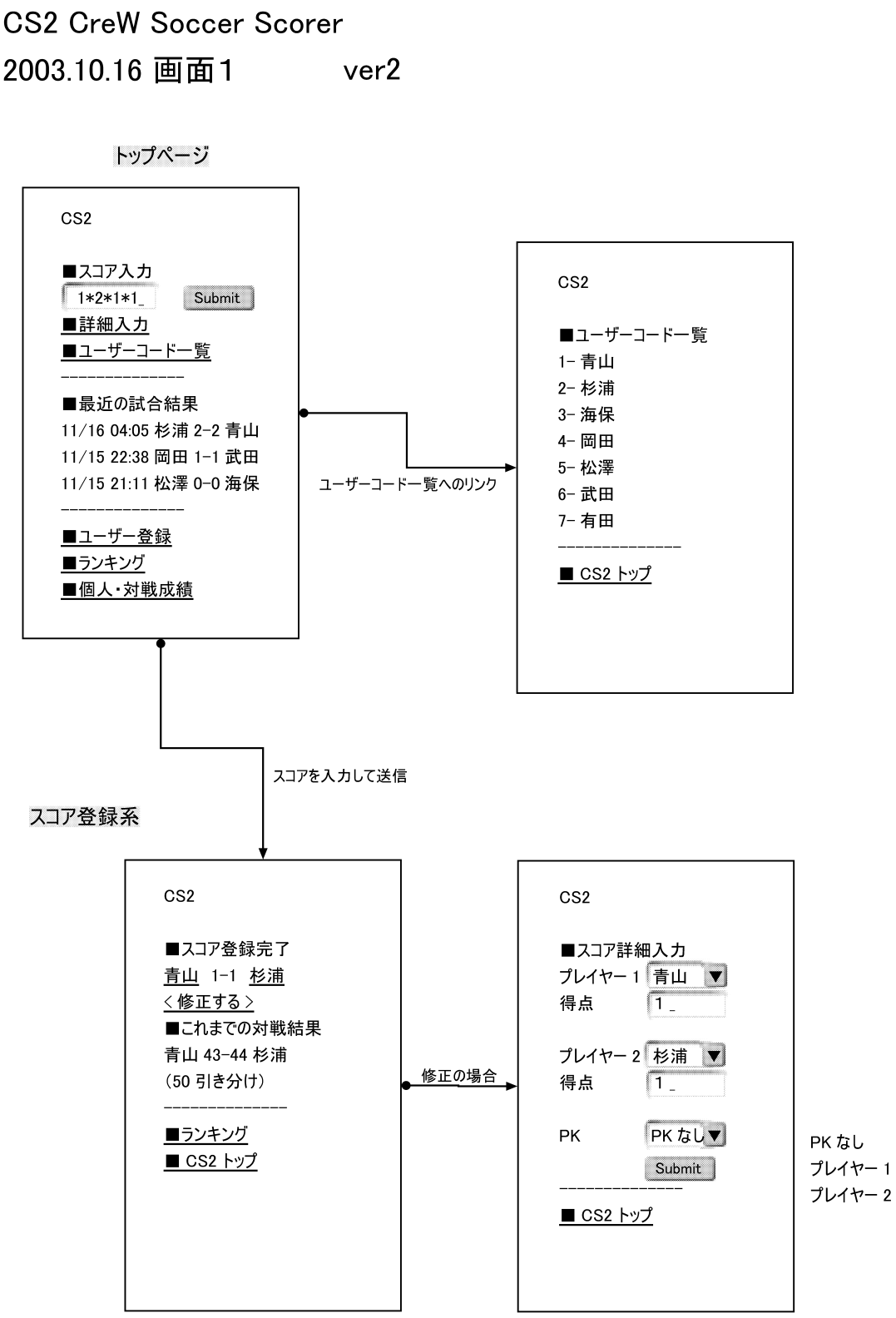


図 1.17: 画面遷移図 ver2 (1)

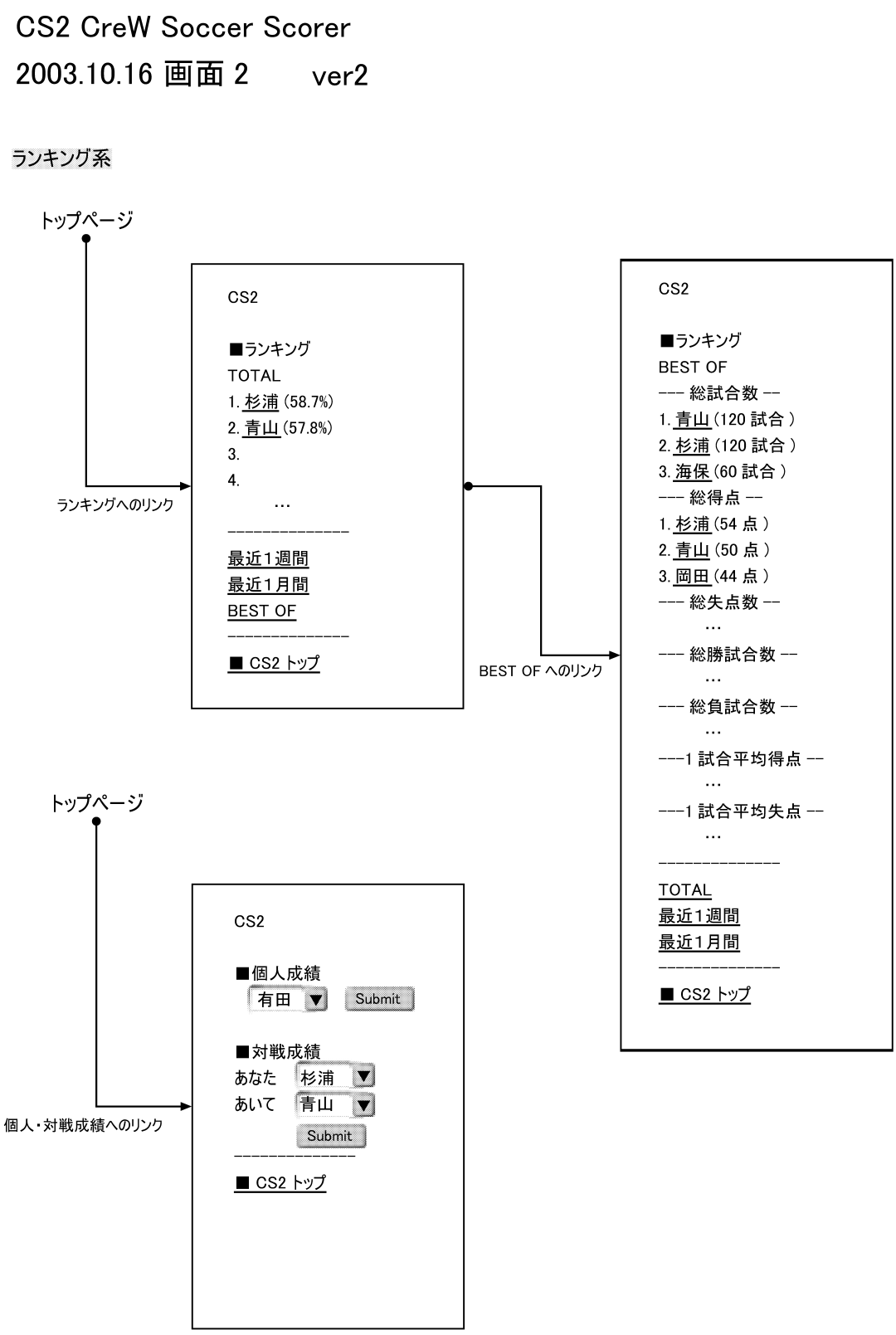


図 1.18: 画面遷移図 ver2 (2)

CS2 CreW Soccer Scorer
2003.10.16 画面 3　　ver2

ユーザー登録系

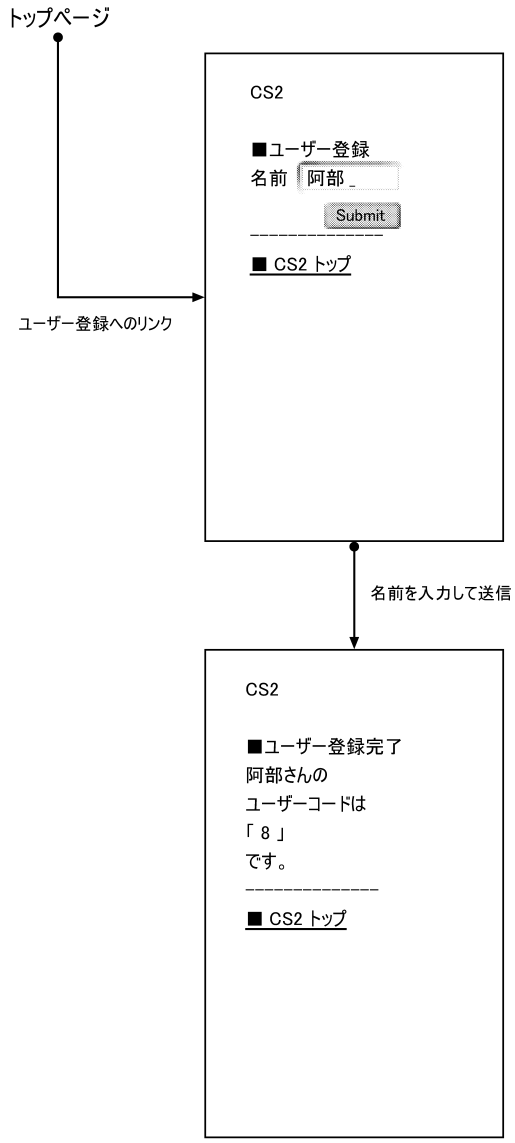


図 1.19: 画面遷移図 ver2 (3)

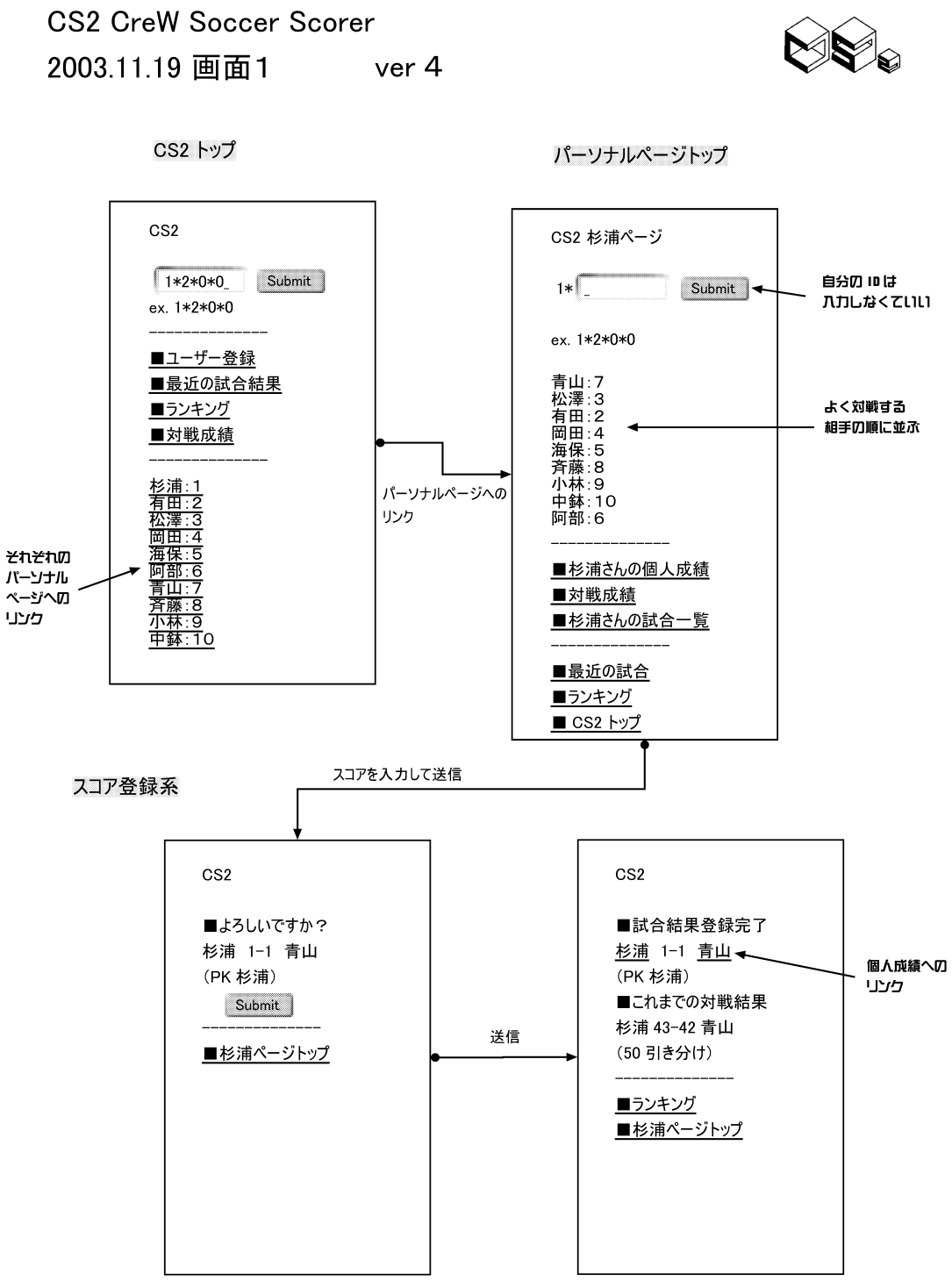


図 1.20: 画面遷移図 ver4 (1)

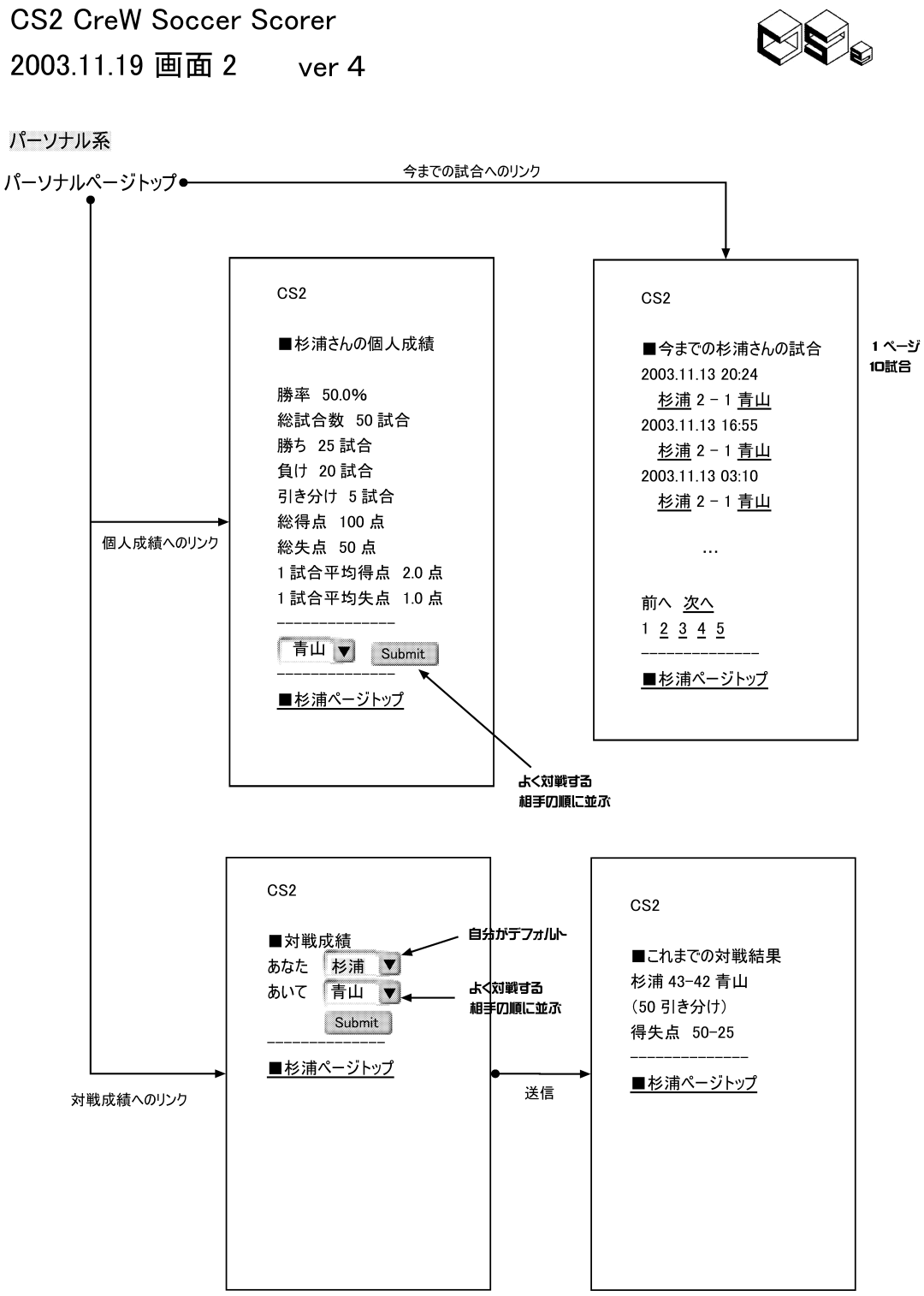


図 1.21: 画面遷移図 ver4 (2)

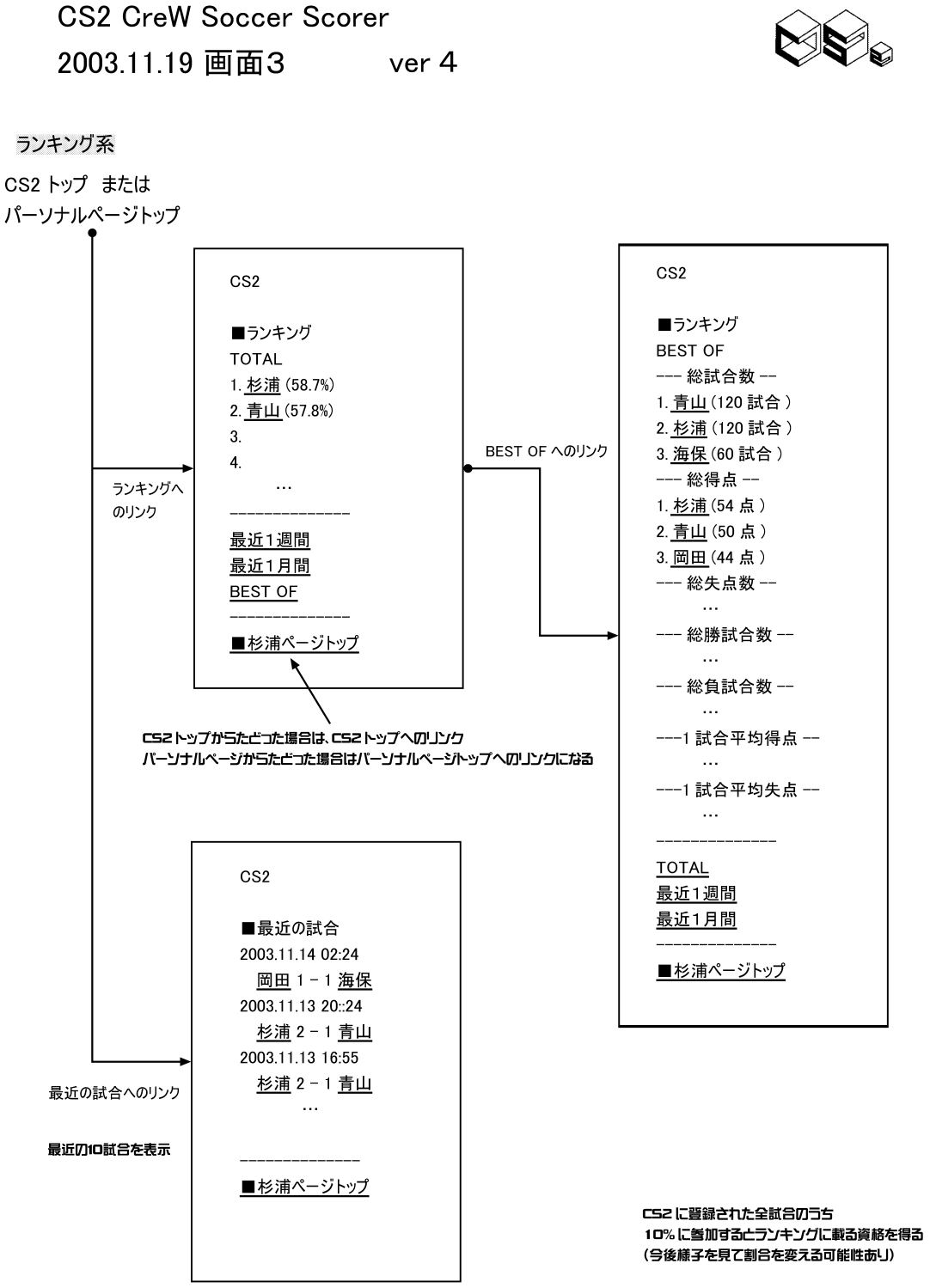


図 1.22: 画面遷移図 ver4 (3)

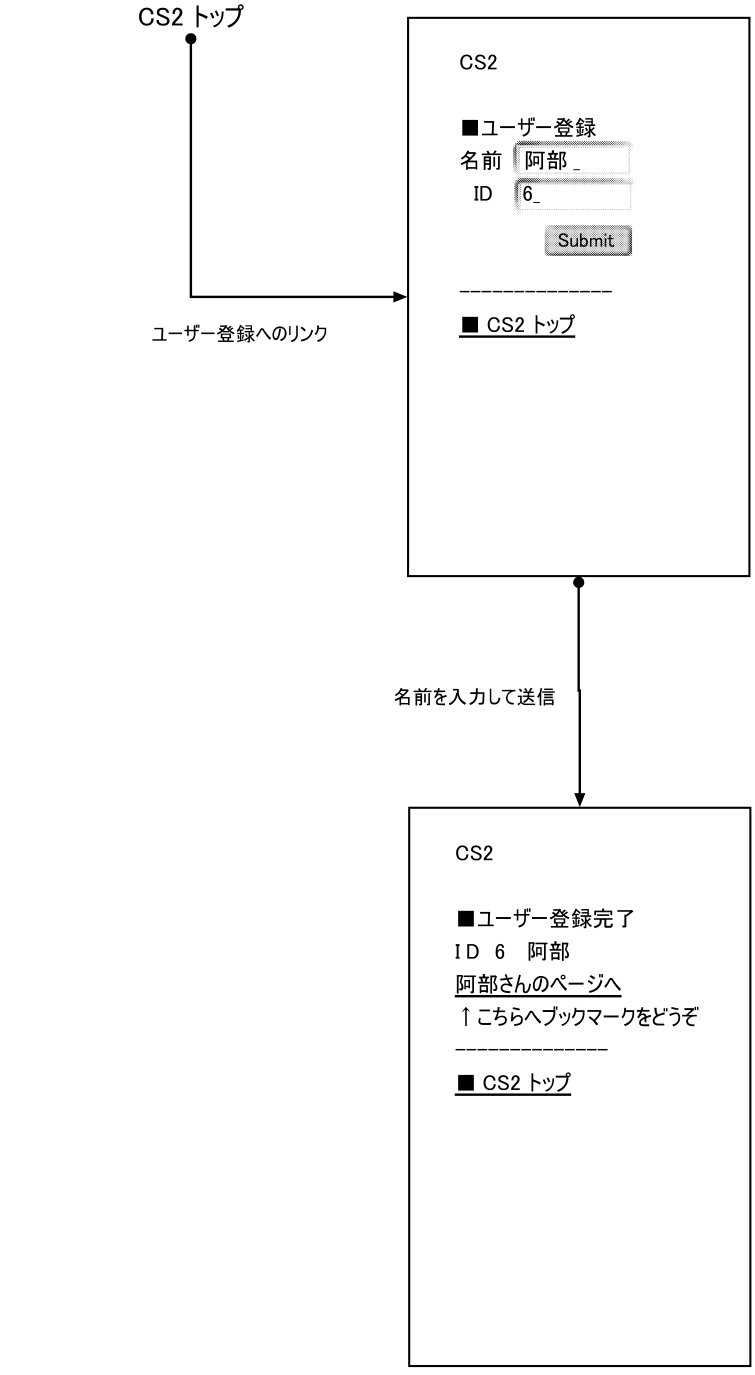


図 1.23: 画面遷移図 ver4 (4)

第 2 章

設計

2.1 概念モデル

CS2 の機能をどのように実現するかを決定するため概念モデルを作成した。

2.1.1 シナリオによる概念モデルの作成

概念モデルを作成するに当たって、PM 杉浦は CS2 が実際に使われている状況を想定したシナリオを作った。わたしたちはそのシナリオを表現するインスタンス図をそれぞれ考えて作成した。(図 2.1 図 2.2)

それらをもとに、システムをどのように設計するか議論した。

リスト 2.1: シナリオ

```
全試合数 4 試合
  2003/10/10 青山 1-0 杉浦
  2003/10/13 青山 2-3 杉浦
  2003/10/14 松澤 3-3 杉浦
  2003/10/14 海保 1-2 ターキー
```

それぞれが書いてきたインスタンス図をもとに議論をした結果、クラスは「プレイヤー」と「試合」のみのシンプルなものになった。(図 2.3)

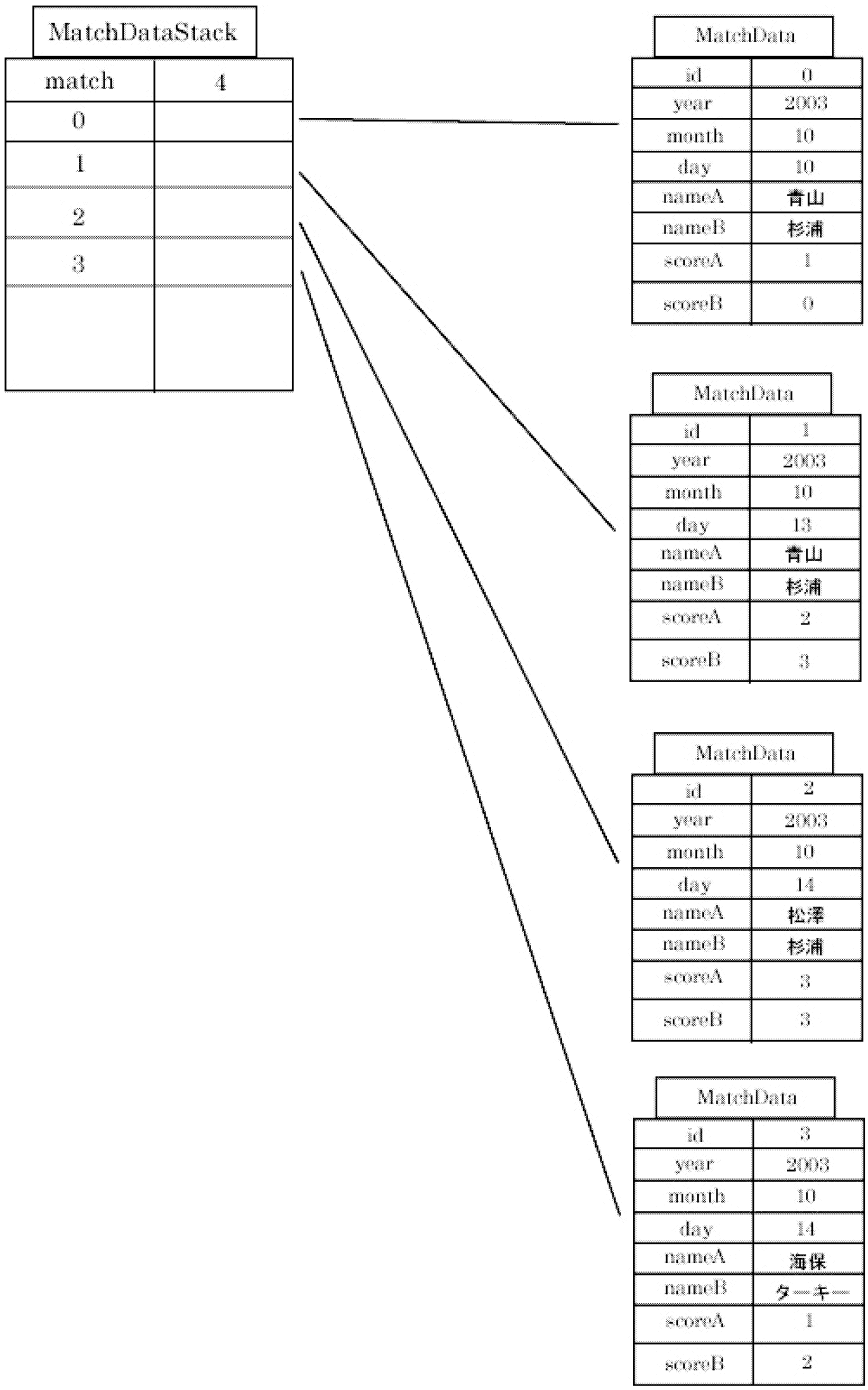


図 2.1: インスタンス図（有田）

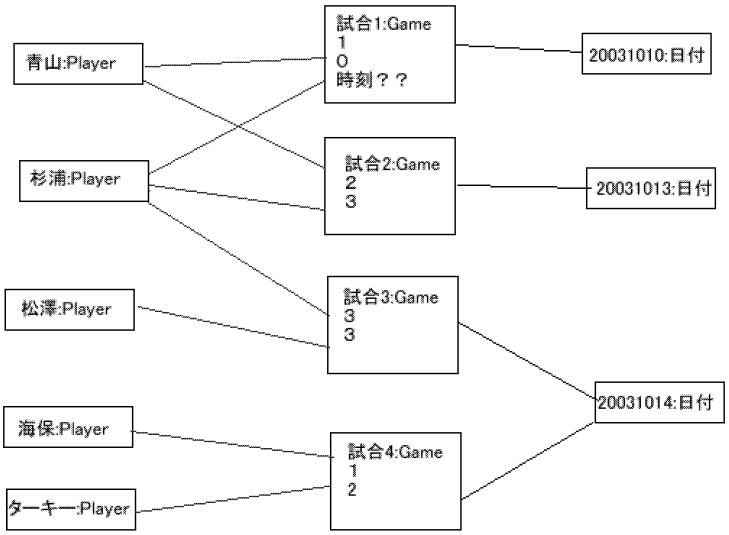


図 2.2: インスタンス図（阿部）

2003.10.30 概念モデル

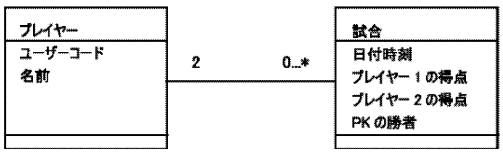


図 2.3: 初期概念モデル図

2.1.2　得点クラスの追加

初期概念モデル図を研究会にて発表したところ、得点を別のクラスにしたほうがいいのではないかという意見がでた。初期概念モデルはシンプルでわかりやすい構成だが、試合オブジェクトの中にプレイヤー A の得点とプレイヤー B の得点があるためにどちらの得点がどのプレイヤーのものか明示的にならず、オブジェクト指向の利点を生かせていない。得点を別のクラスにわけると、所有関係を関連で表すことが出来る。また、サッカーの前半の得点と後半の得点を分けて扱うように拡張する場合に意味のまとまりが明瞭で変更しやすくなるという利点が生まれる。

2.2　プロトタイピング（CUI）

2.2.1　設計モデルとプロトタイピング

普通ならばプロトタイピングを始める前に詳しい設計モデルを記述すべきだが、今回は早めにプロトタイピングを行い、動きを把握してから、もう一度モデルを再考していく手法をとった。これは、本プロジェクトはモデルが非常にシンプルでリモデルしやすいことと、私たちの技術レベルからいって、1 度で運用まで出来るモデルを設計することが難しいためである。

阿部は初期概念モデルに得点クラスを加えたものをもとに参照を加えて、図 2.4 のように CUI プロトタイピングをはじめた。

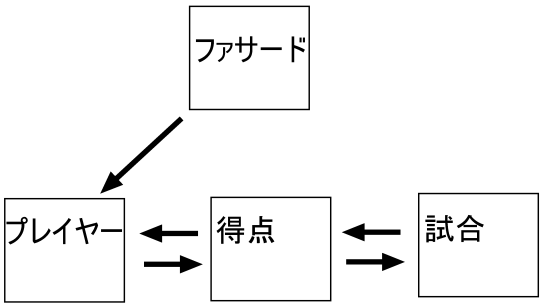


図 2.4: 初期の設計モデル

このモデルで CS2 の基本機能を実装してみると、非合理的な参照のたどり方をしているように感じた。そこで阿部は、どのような関連と参照方向がふさわしいのか紙に書き出した。その際、2 重参照、関連の数をできるだけ減らすこと、参照のつながりが直感的にわかりやすいこと、この 2 つを考慮した。結果、最終的に図 2.5 のようになった。 また、PK の結果も意味の面から得点クラスの管理とした。

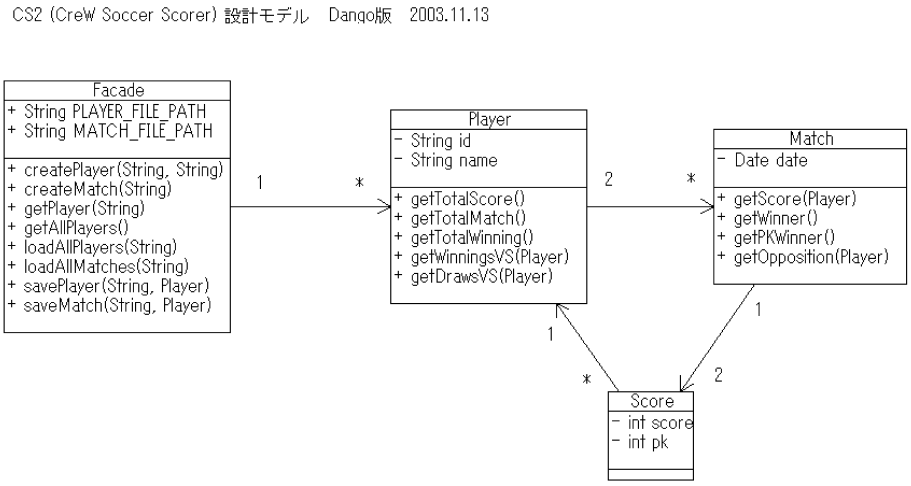


図 2.5: 最終的な設計モデル

2.2.2 CUI プロトタイピングの利点

はじめから WEB アプリケーションを実装するのではなく、CUI から始めることのメリットは、サーブレットや JSP の使用によって起こる無用のエラーに煩わされないことである。CUI でシンプルにプロトタイピングをすることで、モデルの実装に集中することができ、設計の問題点に気がつきやすい。また、プロトタイピング後の本実装においても、機能の追加が発生した場合に WEB を作る前にすぐテストを行うことができ開発が円滑に進んだ。

コラム：「HCP チャートによる設計」

CUI を実装していると幾度となく壁にぶつかり、作業を中止せざるを得ない状況に陥ることがあった。これはシステムの全体の流れを理解できていないことが大きな原因であった。

そこでシステム全体の流れを考えて把握するために HCP チャートを作成することを PM 杉浦に勧められた。まず現在書いているソースコードのコメントを参考にして HCP チャートを作成した。しかしこれはコメント文を並べただけで、システムの流れを読み取ることはできないものだった。

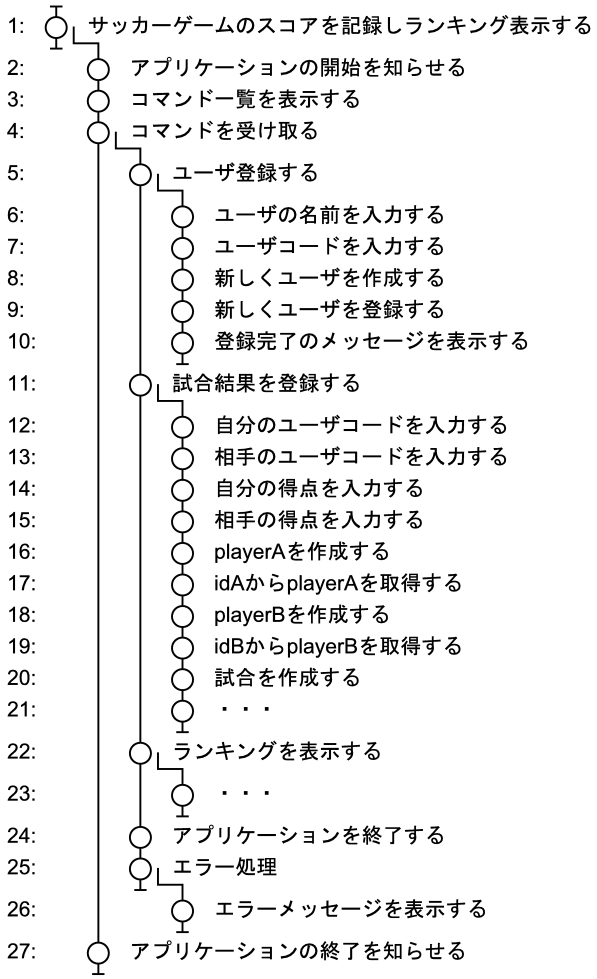


図 2.6: レビュー前の HCP チャート

この段階でレビューをしてもらい、 HCP チャートは前処理、本処理、後処理の大きな３つの流れにまず分けて、そして「試合結果を登録する」等のいくつもある小さな目的の中身を入力、処理、出力の３つの流れに分けてまとめるということを教わった。これらのことに注意して HCP チャートを修正した。

HCP チャートを作ることにより、実装している時に自分が何をしているかが明確になった。もしもわからなければどこがわからないかも把握することができた。何がわからないのかがわからないという無限ループに陥ることもなくなった。その結果、ソースコードに関する質問がより具体的に、わかりやすく、簡潔に伝えることができるようになったのだ。

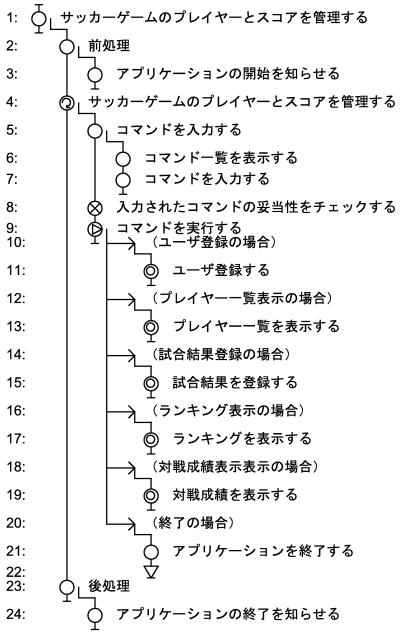


図 2.7: レビュー後の HCP チャート (Main)

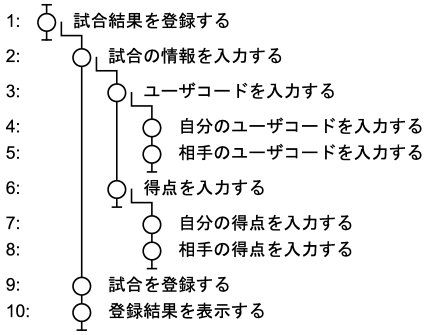


図 2.8: レビュー後の HCP チャート (RegisterMatch)

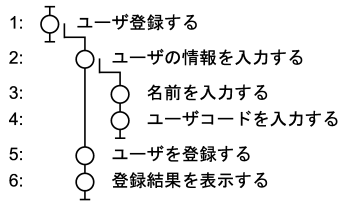


図 2.9: レビュー後の HCP チャート (RegisterPlayer)

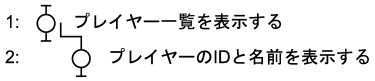


図 2.10: レビュー後の HCP チャート (ShowPlayer)

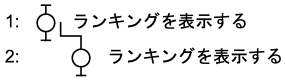


図 2.11: レビュー後の HCP チャート (ShowRanking)

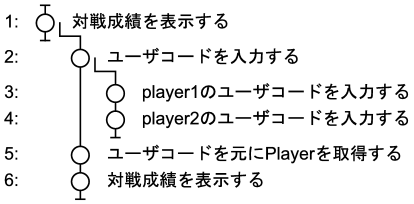


図 2.12: レビュー後の HCP チャート (ShowVS)

第 3 章

実装

3.1 実装

3.1.1 開発環境の統一

グループ開発において環境を統一することは作業を円滑に進めるためにも必須なことだ。なぜならばもしも途中で何か不具合が起きたときに、環境が違ってしまっていてはその原因を究明することが困難となってしまうからである。

3.1.1.1 ソースコード規約

チームでソースコードを書いていく時、ソースコード規約をしっかりと決めることは大切である。以下の規約は杉浦氏が決めたものだ。

リスト 3.1: ソースコード規約

```
 1: package cs2.template;
 2:
 3: import java.util.ArrayList;
 4: import java.util.List;
 5:
 6: /**
 7:  * ソースコード規約 Class
 8:  * →クラスの名前を書きます
 9:  *
10:  * ソースコードの規約を説明するためのサンプルクラスです。
11:  * →クラスの説明を書きます
12:  *
13:  * @author Manabu Sugiura
14:  * →@author と作者の名前を書きます
15:  * @version $Id: SourceTemplate.java,v 1.1 2004/01/27 11:43:33 gackt Exp $
16:  * →@version と CVS タグを書きます
17:  *
18:  * Copyright 2003 CS2 Project All rights reserved
19:  */
```

```
20: public class SourceTemplate {
21:
22:   /*******************************
23:    * 定数
24:    *******************************/
25:   public static final int 定数その１ = 0;
26:   public static final int 定数その２ = 1;
27:
28:   /*******************************
29:    * インスタンス変数
30:    *******************************/
31:   private List 変数その１;
32:   private String 変数その２;
33:
34:   /*******************************
35:    * コンストラクタ
36:    *******************************/
37:   public SourceTemplate(String 変数その２) {
38:     this.変数その１ = new ArrayList(); //インスタンス変数はコンストラクタで初期化
39:     this.変数その２ = 変数その２;
40:   }
41:
42:   /**
43:    * メソッドの目的
44:    */
45:   public void メソッド() {
46:   }
47:
48:   /**
49:    * 戻り値なしメソッドの目的
50:    *
51:    * @param 引数 引数の説明
52:    */
53:   public void 戻り値なしメソッド(String 引数) {
54:   }
55:
56:   /**
57:    * 戻り値ありメソッドの目的
58:    *
59:    * @param 引数 引数の説明
60:    * @return 戻り値の説明
61:    */
62:   public int 戻り値ありメソッド(String 引数) {
63:     return 0;
64:   }
65:
66:   //メソッド間にラベルを仕様する場合は以下を使用すること
67:   /*******************************
68:    * ～関連
69:    *******************************/
70: }
```

3.1.1.2 使用ソフトウェアについて

- java、1.4.1_02
- Eclipse、2.1.1
- Tomcat、4.1

不具合はほとんど出ることなく、順調に作業を進めることができた。

3.1.2 阿部の実装

私が作業した部分に関する説明と学んだところや苦労したところを紹介する。今回のプロジェクトでは、「実装して、ユーザーの意見を聞き、改善」というプロセスを繰り返していたために話が前後しているところもある。

3.1.2.1 暫定版サーブレットの作成

システムの詳細な設計を決める前に、入力を受け付けてファイルに記録するだけのサーブレットを作成した。これは、試合結果を入力するという作業をユーザーに早くから慣れてもらうためと、システムが完全にできてから入力をはじめるのではユーザーの意見を反映するのに時間がかかるので、データを先に集めてシステムの設計に生かすためである。

3.1.2.2 モデル

プロトタイピングのモデルがそのまま本実装に生かされる形となり、それにあわせて整備を行った。その中で苦労したことは、ファイルの読み込み と PK の扱い（PK の入力がなされていない場合と、PK の入力が間違っている場合の区別） などがあげられる。

3.1.2.3 サーブレットの一部

過去のプロジェクトで取り組んだ技術であり、新たに学ぶことは少なかったが、試合結果を登録する際、「入力後、確認ページを表示してから登録する」か、「すぐに登録してしまってから、後で取り消す」かなど、遷移について悩むところはあった。また、PC 版とモバイル版を分離する作業を行った。

3.1.2.4 その他、独自に取り組んだ調査等

勝率計算 このシステムに求められる重要なこととして「公平さ」があげられると思う。「引き分けたら勝率が下がった」というユーザーの声から勝率の計算方法について

調べてみると、様々な算出方法があり、「公平さ」を求めて議論なされていることがわかった。算出方法は大まかに 3 つに分けられる。

1. **勝ち/（勝ち＋負け＋引き分け）**:当初の算出方法。引き分けは負けと同じ。
2. **勝ち/（勝ち＋負け）**:引き分けは無効な試合とする。引き分けが多い人のほうが有利になる。
3. **（勝ち＋引き分け ×0.5）/（勝ち＋負け＋引き分け）**:引き分けを 0.5 勝とする。引き分けると勝率が上がる場合と下がる場合が存在し、わかりにくい。

研究会の発表では 3 の算出法を提案したが、結果的に 2 のわかりやすい案が採用された。

パケット代　CS2 は手軽に試合結果を入力できるように、携帯端末での入力が想定された。しかし、PC でインターネットに接続するのに比べ、携帯端末の料金は割高である。ユーザーの不安を解消するために携帯端末での料金を概算した。

リスト 3.2: パケット代調査結果

```
ページ A　：画像あり　1679 バイト
ページ B　：画像なし　927 バイト
（ともに入力フォームがあるだけの必要最低限のファイル）

ezweb 1 パケット（128 バイト） 0.27 円
  A  3.54 円
  B  1.96 円

docomo 1 パケット（128 バイト） 0.3 円
  A  3.94 円
  B  2.17 円
```

3.1.3　有田の実装

今回私が主に担当した箇所について解説する。

3.1.3.1　ランキング

モデルに書かれているメソッドを用いて、プロジェクト内で決めた種類のランキングを表示できるようにした。一度作ってそれで終わりではなく、毎週ある研究会の進捗報告でユーザーから出た要望に応えるべく、何度も何度も改良を繰り返した。例えば、「最近一週間のランキング」とは具体的にいつからいつまでを示しているのかがわからないというユーザーの不満を解消すべく、すぐにその日付を取得し表示できるようにした。また、1 試合だけしかしていないプレイヤーがその試合に勝っていたために、いつまでもランキン

グ 1 位に君臨するという問題が生じたときも、ある一定の試合数をこなしていないプレイヤーはランキングに表示されないようにすることで改善した。ユーザーの要望すべてに応えるのではなく、本当に必要なことなのか、改善すべきことなのかをプロジェクト内でしっかり話し合うことがとても大切だと感じた。

3.1.3.2 ブラウザ版のデザイン

モバイル版とブラウザ版の両方を作ることが決定した時、ブラウザ版のデザインを私がやることになった。作業内容はインターフェイスデザインの決定、html によるテンプレートの作成、テンプレートを参考にしながらの jsp 書き換え、であった。インターフェイスデザインは手書きで紙に書いてメンバーに見せ、そのデザインで良いか検討してもらった。そのデザイン画を元にして、トップページ、パーソナルページ、その他のページを簡単に html にて作成した。ここではレイアウトと配色に気を使った。1 つの色を決めるのに 2 時間以上かけることもあった。テンプレートが完成して、そのレイアウトになるように jsp をすべて書き換えていった。書き換える作業はそれほど大変ではなく、むしろ見栄えを良くするための画像の作成などに時間をとられた。いつも思うことだが、インターフェイスはわかりやすく使いやすいことを最も重要視してデザインすべきである。

コラム：「サッカーに関する英語について」

サッカーに関する用語などは辞書ではわからないことが多く、きちんと調べたほうがよいということを学んだ。

CS2 を設計する際、クラス名ははじめ日本語で作られた。これは、私たちは日本人であるため、当然日本語であるほうが、メンバーの共通理解が得やすく、無用な行き違いを防ぐことが出来るためである。そして、私たちはコーディングするにあたって、日本語のクラス名を英訳した。それは当初次のようであった。

- 試合クラス→ Match
- プレイヤークラス→ Player
- 得点クラス→ Score

ここで問題だったことは、この英訳は和英辞書から得られたものであり、実際にサッカーにおいてどのような英単語が使われているかきちんとしらべなかったことである。

その問題点が明らかになったのは、ランキングを作成することになってからだった。ランキングは、一般的にプロ野球順位を表すのに使うような、チームごとの試合数、勝利数、勝率などを示した表を使う。「総得点」を和英辞書で調べるとそのままでは辞書に載っていない。そのためとりあえず、総＋得点で Total + Score としていたが、あとになってソースコードを見直す際にあまりにもいいかげんであるので詳しく調べた。アメリカやイギリスのサッカーのホームページを見ると、GP、GA など、省略されていて何を表しているのかよくわからなかったがこれをさらにしらべると、次のようだった。

総試合数　GP(Games Played)
勝利数　W(Wins)
敗北数　L(Losses)
総得点　GF(Goals For)
総失点　GA(Games Against)

このことから、サッカーの試合は一般的に match でなく game が使われること, 得点には goal が使われることがわかった。しかし、この時点で、基幹となる部分のクラス名を変更することは混乱を招くので避けた。ランキングの部分だけ GF や GA といった略称で使用した。

参考

- アメリカ Yahoo! sports[*1]
- サッカーで使われる英語[*2]

[*1] http://sports.yahoo.com/fbit
[*2] http://www.japantimes.co.jp/shukan-st/sports/ss-terms.htm

コラム：「CSV ファイルを読み込むオートマトン」

テキストファイルの読みこみはどのシステムにおいてもよくあるが、状態遷移図を書くことで文字列のパースを理解し、コーディングを助けた

CS2 の試合結果は、「1*2*1*1」といったコードで登録される。この場合は「ID が 1 のプレイヤーと ID が 2 のプレイヤーが試合をして得点が 1 対 1 であった」という意味になる。この結果をシステムが内部でファイルに書き出す際、保存のフォーマットを CSV ファイル形式で「”日付時刻”,”プレイヤー１”,”プレイヤー２”,”プレイヤー１の得点”,”プレイヤー２の得点”」とした。先の例だと「”2004/01/01 15:47:21”,”1”,”2”,”1”,”1”」といったようになる。このファイルを今度は読み込むときに、どうやってデータの部分だけうまく格納するかということが問題になった。PM のアドバイスによりまず、1 文字ずつ読み込んだときどのように状態が変化するか、紙に書いた。(図 3.1) また、この図からファイルを読み込むためのコードを書くことが出来た。

CSV ファイルを読み込むための状態遷移図 2003.11.10 Dango

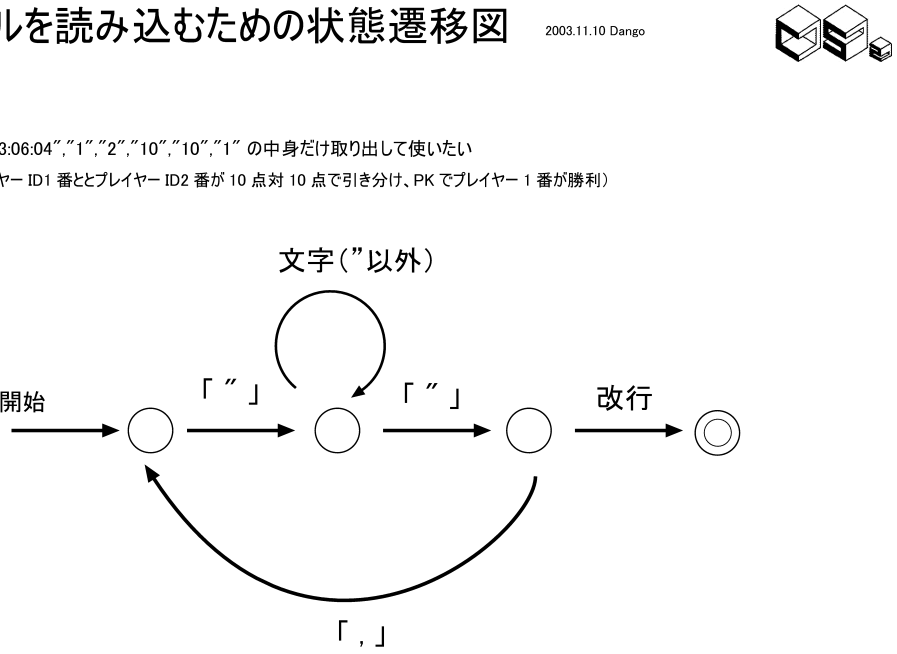


図 3.1: CSV ファイルを読み込むオートマトン

リスト 3.3: CSV ファイルを読み込むクラス

```
1: package cs2.util;
2:
3: import java.util.ArrayList;
4: import java.util.List;
5:
6: /**
7:  * CSV フォーマットのファイルを読むクラス
8:  *
```

```
 9:  * @author Dango
10:  * @version $Id: File.java,v 1.2 2004/01/24 12:38:47 dango Exp $
11:  *
12:  * Copyright 2003 CS2 Project All rights reserved
13:  */
14: public class File {
15:
16:   private static int STATE_A = 10; //単語を読む前の状態
17:   private static int STATE_B = 20; //単語を読んでいる状態
18:   private static int STATE_C = 30; //単語を読み終わった状態（カンマの前）
19:
20:   private int state; //状態
21:   private String word; //単語
22:   private List wordList; //単語のリスト
23:
24:   /**
25:    * コンストラクタ
26:    */
27:   private File() {
28:     state = STATE_A;
29:     wordList = new ArrayList();
30:     word = "";
31:   }
32:
33:   /**
34:    * CSV フォーマットの文字列を分解する
35:    *
36:    * @param line CSV フォーマットの文字列
37:    * @return 単語（内容）の配列
38:    */
39:   public static String[] decode(String code) {
40:     File file = new File();
41:
42:     for (int i = 0; i < code.length(); i++) {
43:       if (file.state == STATE_A) {
44:         file.checkA(code.charAt(i));
45:       } else if (file.state == STATE_B) {
46:         file.checkB(code.charAt(i));
47:       } else if (file.state == STATE_C) {
48:         file.checkC(code.charAt(i));
49:       }
50:     }
51:
52:     return (String[])file.wordList.toArray(new String[file.wordList.size()]);
53:   }
54:
55:   /**
56:    * 状態 A の処理をする
57:    *
58:    * @param c 文字
59:    */
60:   private void checkA(char c) {
61:     if (c == '"') { //ダブルクォーテーションを読み込んだら
62:       state = STATE_B; //状態 B へ
```

```
63:     }
64:   }
65:
66:   /**
67:    * 状態 B の処理をする
68:    *
69:    * @param c 文字
70:    */
71:   private void checkB(char c) {
72:     if (c == '"') { //ダブルクォーテーションを読み込んだら
73:       state = STATE_C; //状態 C へ
74:       wordList.add(word); //単語をリストに追加
75:       word = ""; //単語を初期化
76:     } else {
77:       word = word.concat(Character.toString(c)); //単語に文字をつなげる
78:     }
79:   }
80:
81:   /**
82:    * 状態 C の処理をする
83:    * @param c 文字
84:    */
85:   private void checkC(char c) {
86:     if (c == ',') { //カンマを読み込んだら
87:       state = STATE_A; //状態 A へ
88:     }
89:   }
90: }
```

第 4 章

評価

4.1 評価

4.1.1 入力のテスト

CS2 を WEB で公開するにあたって入力のテストを行った。

リスト 4.1: テストする入力文字列のリスト

```
正常
  1*2*1*2
  1*4*4*5*4  //PK の結果あり

異常 (1)
  "1"*7*7*9    //関係ない記号が紛れ込んでいる
  ?*?*?*? //数字でない文字が入力されている
  1*7*1*1*う    //数字でない文字が入力されている
  1*4*4*う       //数字でない文字が入力されている

異常 (2)
  1*7*1*0*1    //引き分けでないのに PK の結果が入力されている
  1*7*1*0*4    //PK の勝者に関係ないプレイヤー ID が入力されている
  1*1*1*1       //同一のプレイヤーが入力されている
  1*1*1*1*8  //同一のプレイヤーかつ PK の勝者に関係ないプレイヤー ID が入力されている

異常 (3)
  1*1000*1*0       //プレイヤーリストにないプレイヤー ID
  1*7*1*1*1000  //PK の勝者にプレイヤーリストにないプレイヤー ID が入力されている
```

異常と分類されたものの中で (1) は入力の形式が正しくないものである。(2) は、入力の形式は正しいが、データの内容が論理的に受理されないものである。(3) は、入力の形式が正しく、データの内容は場合によって受理されない場合があるものである。

(2) と (3) はファサード (model) の中で対処するべきものだが、(1) はインターフェース（view）に依存するものであり、サーブレット・JSP 内で対処した。

4.1.2　プロトタイプに対する評価

システムを導入しユーザーテストを繰り返した。 プロトタイプを使ってもらった時からユーザーテストはスタートした。プロトタイプは「試合結果の入力」「ユーザー登録」「対戦成績の表示」 「個人成績の表示」、 以上の機能のみを備えていた。

プロトタイプへの要望

1. プルダウンでの入力のほうが使いやすい
2. 入力画面は入力画面だけのほうがいい
3. プレイヤー ID 表が入力画面の下にあったほうがいい
4. 入力→確認→登録の流れにして欲しい
5. パーソナルページが欲しい（URL 振り分け）
6. 自分の成績が見たい
7. 最近の試合結果が見たい
8. ここ 1 週間の試合結果が見たい
9. ランキングに勝率、失点なども欲しい
10. ID や試合数などをクリックすることで、ランキングをソートできたら嬉しい

これらの評価を受けて、実際に改善するかを論議した結果、1 と 2 以外は全て反映させることに決まった。そしてこれらの機能が一通り揃った段階で改善版としてリリースした。

4.1.3　改善版への評価

改善版への要望

失点ランキングが失点王になっているので、逆にして欲しい　敗北数、総失点、平均失点に関してはその数値が低いプレイヤーが上位にランキングするように変更した。これにより、どのランキングも強いプレイヤーが上位に位置することになった。

勝率はパーセンテージではなく、何割何部何厘にして欲しい　それまでは「33.3%」と表示していたが、それを「3 割 3 分 3 厘」という表示に変更した。しかしプレイヤーはそのような表示ではなく「0.333」のような表示を求めていたのだ。「.333」という表示にしたほうが良いのではないかと考え、変更を試みたが、週間ランキングにおいて勝率が「1.000」になるプレイヤーが必ず出てくるので「0.333」の表示にすることが決定した。

取り消し機能は、日付で一意に特定できるのか不安　取り消し機能の不安に対する改善案はすべての試合に ID を振ることがあげられた。だが登録した時間は秒単位で数えていることから、同時に入力されることはないと判断し、変更はしなかった。

見た目をブラウザ用か携帯用かの住み分けをしっかり工夫 インターフェイスは当初から携帯での使用と前提としてきたためにシンプルなものだった。しかしプレイヤーを見てみると段々とブラウザからの利用に変化していた。特にランキングはブラウザで見たほうが断然見やすいことが要因だと思われる。入力はブラウザで行う人もいれば携帯で行う人もいた。そこでモバイル版とブラウザ版の両方を作成することに決めた。

週間ランキングより、何月のランキング、何月何週のランキングがほしい 毎月のランキングなど一定期間ごとのランキングは作らないことした。

最近の自分の試合が見たい 最近の自分の試合だけでなく、すべての人のすべての試合の記録が見られるようにした。

第 5 章

感想

5.1 阿部の感想

5.1.1 これまで

前回、2003 年度春学期では「みるみる」というプロジェクトに参加した。そこで私の果たした役目というのは、やることを与えられて順々にそれをこなしただけ、といった印象が強く、自分のしたことの意味をあとから理解することも多かった。 そこで、次のプロジェクトでは「自分からやることを考え、実行する」また「自分が学んだことを後輩に伝えていく」これらを目標として考えていた。

5.1.2 わたしの成果と反省

プロジェクトをどう進めていくか、というのは PM の仕事であるとはいえ、前回の経験から、今回は全体の進み具合を把握しながら仕事を進めることができた。他に個人的な成果としては、多少なりとも自信がついたということである。前回のみるみるに比べると、CS2 はシンプルで小規模だが、同学年のメンバーが他にいないので自分がプロジェクトの主力として引っ張っていくという緊張感があったが、それがやりがいや自信につながった。

問題点は 技術レベルの追いついていない有田くんの指導に関して PM に頼ってしまったかな、というところ。また、仕事の時間を記録する進捗ログも後半ではだんだん手を抜いてしまい、自分でいいかげんだったと思う。

5.1.3 プロジェクトに関して

基本的には、研究室の遊びから生まれたプロジェクトなのでリアルタイムに使ってもらうことができ、すぐに感想が飛んできて楽しかった。逆に「研究室の人が使う」という甘えから、エラーの対応に厳しくなれなかったところもあるかもしれない。他には、有田くんとの進み具合の差から、実装での悩みをあまり共有できない寂しさがあったり、自分の

やるところをきれいにわけることができず、どこまでやったよいか悩んだりした。

5.1.4　これから

自分としては CS2 で学んだことを卒業後も少しでも生かしていけるといいなと思う。また、来学期のある有田くん、杉浦さんも、さらに良いプロジェクトができるように次の研究会で是非がんばってほしい。

杉浦さん、見事な PM ぶりありがとうございました。おかげでいい雰囲気で進められました。

有田くん、今考えるとくじけずによくがんばったなぁと思います。なんだかそれだけで感動です。かっこいいデザインありがとう。

CreW の皆様、ご協力大変ありがとうございました。重ねて言うようですが、これからも使ってくださいね。

5.2　有田の感想

プロジェクト候補が発表された時、すぐにこのプロジェクトにしようと決めた。理由はいろいろある。まず自分自身ゲームがとても好きで日頃から遊んでいること。次にサッカーという点で幼い頃からずっとプレーし続けていたこと。そして最後に 実装の難易度が低そうだったこと。

私は実装に苦手意識を持っている。そこで、難しいものに挑戦するよりも簡単なもので基礎を固めるほうが自分のためになると判断したのである。しかし大きな誤算があった。それは私の実装力が想像以上に低かったことだ。「○○の実装をしてください」と課されても悩んだ末、結局わかりませんでした、で終わることが多かった。

考えてみるとこれまでのプログラミングの授業、オブジェクトプログラミング、研究会のサブゼミ、すべての授業に共通して言えることは「あるソースコードを参考にし、それを改造することで実装を進める」方針だったことである。しかし今回、プロジェクトの実装では参考にするソースコードが存在しなく、ある意味、そのすべてを自分で考えなくてならなかった。そしてその力がない私の CS2 プロジェクトの実装への貢献度は非常に乏しかった。

そうは言っても、実装に苦労しながらも身につけたものは多かった。オブジェクトプログラミングのテキストを読み、復習することで、メソッドやクラスやリストの使い方を再確認できたり、新しく JSP を学べたりもした。そして今後最も役に立つであろう HCP チャートに関して勉強できたことは非常に良かったと 思っている。

では実際に私はどのような点でプロジェクトに貢献したのだろうか。私がした大きな仕事は企画書の作成、入力方法提案書の作成、ロゴ作成、ランキングの作成、ブラウザ版のインターフェイス作成。細かい仕事は他にもたくさんあったと思う。

企画書は目的が大きな論点となった。「強さをはっきりさせることでプレイヤー全員が

仲良くなること」。これが私が最初に提案した目的だった。しかし強さをはっきりさせることで仲良くなる保障など存在するはずも無かった。そこでデータを提示し、強さをはっきりさせることだけを目的とした。その後プレイヤー同士の人間関係がどうなろうとCS2 プロジェクトは関知しないという姿勢で挑むことに決定した。これは非常に面白い目的だと私は感じた。CreW は人を幸せにするアプリケーションを作ることを大きな目的としているにも関わらず、それに反した目的を掲げたからである。

入力候補提案は案を考え、議論することがとても楽しかった。ユーザビリティに興味がある私は 4 つの候補をあげ、最も使いやすく現実的なバーコードの入力を推したが、プロトタイプのコード入力が定着していた。確かに専用のマシンからの入力に限定されるよりは、自分の携帯や人の携帯、ブラウザとあらゆる場所から入力できるコード入力のほうが使いやすいとも考えられる。

ロゴは 2 つ作り、進捗報告で発表し選んでもらった。そして残ったほうをより見やすくなるように手を加え、正式なロゴとした。CS2 という言葉を聞けばこのロゴが頭に浮かんでくれるようなデザインを目指した。

ランキング作成への貢献度は大きいと思う。始めにランキングの種類を議論したのだが、ゲーム好きの立場から意見を言い、平均得点などを提案した。また、手集計でランキングを作ることにより、その労力や計算間違いする可能性から人の手で行うべきではないことの判明や、実装のためのシミュレーションができた。実装に移ってからは、進捗報告で出た意見や要望を反映する役割となっていた。

ブラウザ版のデザインはやはり見易さと使い易さを最も重要視した。色彩の選別には数時間も要した。何も説明がなくても、プレイヤーがどのコンテンツへも自由に行き来できるようなら私のデザインは成功であったといえるだろう。

最後に、私の多量の質問に親身になって答えてくださった杉浦さん、阿部さんには感謝し尽くしてもとても足りません。本当にありがとうございました。