Vol.2015-HCI-162 No.17
2015/3/14

作業進捗状況と成果物イメージの共有による グループハッカソンにおける協調活動支援

西 康太郎[1]　西本 一志[2]

概要：本稿では，ハッカソンチームの協調活動のパフォーマンス向上に役立てるために，「成果物イメージ共有を兼ねたプロトタイプ Ver.0 作成機能」および「作業進捗共有を兼ねた画面共有機能」から構成される支援システム「HackathonMediator」を提案する．本システムを実際のハッカソンに導入し，参加メンバーに対して実施した行動観察およびインタビューによる評価の結果を報告する．

Supporting Collaboration in a Group Hackathon by Sharing States of Progress and Product Images

Nishi Kohtaro[1]　Nishimoto Kazushi[2]

Abstract: This paper proposes a support system of a group hackathon named “HackathonMediator” to solve problems of collaboration in a team of hackathon. HackathonMediator consists of two functions: a function to share work screen among members for sharing states of progress and a function to create a mock-up model for sharing product images. We applied this system to an actual Hackathon event. Results of evaluation by behavioral observations and interviews are reported.

1. はじめに

近年，ハッカソン（Hackathon：Hack-a-thon）[*1] と称される，数名で構成されるグループによる短期間での作品制作大会が盛んに実施されている．ハッカソンには，従来の共同開発のスタイルとは大きく異なる特徴がある．第 1 は，従来は回避すべき状態であったデスマーチが前提となる点である．多くのハッカソンにおいて，チームは 1〜2 日程度のハードなスケジュールの中で優れたアイデアを企画し，開発・プレゼンテーションを行い，革新的な成果物を生み出さなければならない．第 2 に，現場で即席に構成されるチームによる，期間限定での密度の高いコラボレーションが求められる点である．しかもチームには，プロフェッショナルやアマチュア，プログラマやデザイナなど，様々な技能や技量を持つ人が入り混じる．彼らの日常業務における常識は，ハッカソンチームにおいて通用しない．

このような特徴を持つため，成果物を実現するに至らないケースや，チームが途中で空中分解してしまうようなケースなど，失敗に終わる事例が多数見受けられる．今後ハッカソンがさらなる発展を遂げ，参加者と作品としての実績を増加させるためには，このような失敗を回避する手段を実現することが望まれる．我々は，チーム内での情報共有が成功のための重要な鍵となると考えた．

そこで本稿では，ハッカソン状況下での協調型開発において，チーム内での情報共有に特化した手段を提供するシステム “HackathonMediator” を提案する．HackathonMediator は，「成果物イメージ共有を兼ねたプロトタイプ Ver.0 作成機能」と，「作業進捗共有を兼ねた画面共有機能」の 2 つの機能で構成される．ハッカソンの企画立案終了時に「成果物イメージ共有を兼ねたプロトタイプ Ver.0 作成機能」を活用して，統一されたプレゼンテーションモデル（モックアップ）を構築する．これにより，各チームメン

[1] 北陸先端科学技術大学院大学 知識科学研究科
School of Knowledge Science, Japan Advanced Institute of Science and Technology
[2] 北陸先端科学技術大学院大学 ライフスタイルデザイン研究センター
Research Center for Innovative Lifestyle Design, Japan Advanced Institute of Science and Technology

[*1] ハッカソンとは “Hack” と “Marathon” からなる造語である．

バが思い描くプロダクト像の差異を吸収する．しかも，この機能を用いて作成されたモックアップは，実際のアプリケーションのソースコードとして直接活用できるため，企画段階から開発段階へと効率的な移行を可能とする．続いて開発過程で，「作業進捗共有を兼ねた画面共有機能」を活用することにより，各チームメンバは他メンバの作業状況を遅延なく把握しつつ作業を進めることができる．この機能では，各メンバが特定の作業（アプリケーションのデバッグ，画像作成，プレゼンテーション資料作成など）を行うタイミングに合わせ，該当メンバの作業画面を大型ディスプレイにミラーリングすることにより，各メンバの作業状況をチームが効率的に確認できる機会を提供する．

2. 関連研究

プロジェクトメンバ同士のコミュニケーションログの可視化 [1] などをはじめ，組織内における情報共有の支援は古くから研究されている分野である．さらに，Agile を筆頭として，小規模の組織での協調型開発に焦点を当てたプロセスモデルの研究開発も近年増加している [2][3][4]．しかしながら従来の研究は，長期的なスケジュールの中で，企業など確立された組織での協調型開発を前提としたものがほとんどであり，デスマーチや期間限定のコラボレーションが想定されるハッカソンへの導入に焦点を当てた研究事例は見受けられない．

ハッカソンおよび同様の形態を持つイベントに関する調査報告は，近年少しずつ増加してきている [5][6][7][8]．しかし，従来の調査報告のほとんどはイベント全体の運営に焦点があてられており，各ハッカソンチームの内部で起きていた事象について調査している事例は見当たらない．

3. 予備実験

HackathonMediator を開発するにあたって，チームの失敗要因およびその解決手法を，チームの内部で起きている問題に注目して検討するための予備実験を行った．予備実験では，ハッカソンを 4 度実施した．参加メンバは，実験毎に異なる．各実験において，チームの行動を観察し，失敗要因として考えられる問題を洗い出した．また，2 回目以降の実験では，それ以前の実験で確認された失敗要因に基づいて考案した解決手法を導入した．各実験において著者は，ハッカソンの運営者として振る舞い，実験中に気になった行動などについては，随時インタビューを行った．運営者に対し，参加者が立案した企画に関する感想や意見を求めたり，技術に関する相談をすることは許可した．ただし，チームマネジメントに関して運営側が意見を述べることは禁止した．

3.1 2014 年 4 月 19 日 8 時間ハッカソン

初回の実験では，主催側からの指示は行わず，自由に作業を進めてもらう形で進行した．主な問題として，作業が進むにつれてチームがコミュニケーションをとらなくなる様子が確認された．各チームメイトが担当するタスクの内容や進捗が次第に共有されなくなったほか，共有された情報に他メンバが気づかなかったり，放置したりする例も見受けられ，誰が何をやっているのか把握できない状態に陥っていたことが確認された．

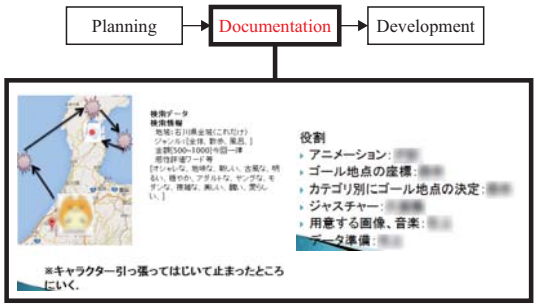

図 1 プレゼンテーション資料の一部

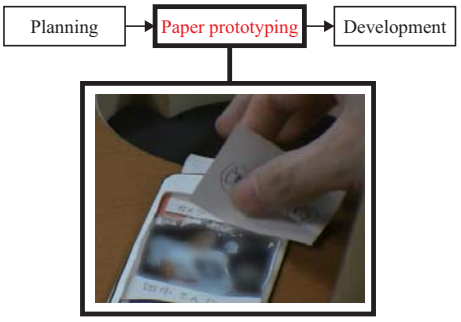

図 2 ペーパープロトタイピングの様子

3.2 2014 年 5 月 31 日 8 時間ハッカソン

2 度目の実験では，企画立案終了時に，まずプレゼンテーション資料を作ることで事前の情報共有を図った．作成する資料には，各々が担当する作業や成果物の完成イメージなどを記述するように促した（図 1）．この手法において確認された問題としては，「モノができていく過程でようやく資料の内容を理解した」，「開発に残された時間を気にしていて，資料作りに関わらなかった」などの旨が報告され，資料内容の認識に差異が発生したことが判明している．また，このチームでは互いの作業に関与し合う場面が少なく，個人での作業に徹した結果，リソースの受け渡しの際に相手を間違えるなど，作業分担に対する把握不足が見られた．

3.3 2014 年 7 月 19 日 10 時間ハッカソン

資料に対する認識の差異を解決する上で，3 度目の実験では個人別ペーパープロトタイピングを導入した（図 2）．企画立案の終了時，各メンバが成果物に抱く画面構成や動作イメージをペーパープロトタイプとしてそれぞれ表出化し，それらを互いに見せ合うことによって精度の高い情報共有を図った．このプロセスにより，各メンバが成果物に

抱くイメージの差異（画面順序の違い，機能数の違いなど）が表出化され，それらの差異を吸収する作用が見受けられた．この手法で確認された問題としては，開発過程において「実現が困難な機能」の切り捨てなどが発生したことで，作業の優先順位が判別できなくなり，成果物のコンセプトが徐々に崩壊していく様子が確認された．このことから，機能の切り捨てが発生した際などに，チームに迅速な認知や意思決定を促すためのタイムリーな情報共有が必要であると考えられた．

3.4 2014 年 8 月 2-3 日 30 時間ハッカソン

4 度目の実験では，参加メンバ全員の作業画面を大型ディスプレイ上で常時共有する手法をとった（図 3）．大型ディスプレイ上では，各メンバの作業画面がグリットレイアウト形式で表示されており，全メンバの作業状況を一望できる．この手法により「みんながどんな行動をとっているか分かった」といった感想が得られたが，時間の経過と共に共有画面に対する注目頻度は大きく減少した．このことから，作業画面を常時共有するだけでは効果が薄く，重要な場面だけをフィルタリングし，確実に共有する仕組みが必要だと考えられた．

3.5 確認された問題

予備実験を経て確認された問題を以下に列挙する．

(1) **開発への焦り**：企画段階での議論が不十分なまま開発へ移行することについて，あるメンバからは開発に残された時間を意識した上での行動であることが示唆された．

(2) **他メンバへの関心の希薄化**：開発過程において各メンバが作業に追われた結果，チーム内で互いの作業に関与し合う場面が少なくなっていく様子が確認された．

(3) **情報共有の頻度低下**：インタビューで，「作業が進むにつれ周囲も忙しくなり，どこまでの情報を共有して良いか分からなくなった」などの意見が述べられている．

(4) **共有された情報の確認不足**：ハッカソンの最中に共有された情報について，「作業に夢中で気づかなかった」，「何かが共有されていたことには気づいていたが，先に作業を済ませてから確認しようと思った」などの意見が述べられた．

(5) **進捗状況の把握不足**：複数のチームのメンバが「誰が何をやっているか分からない」状態に陥ったことを述べた．

(6) **作業内容の重複**：複数のチームにて，担当メンバ・担当でないメンバが同一の作業に着手していたことが確認された．

(7) **成果物に抱くイメージの差異**：プレゼンテーション資料内に表記された一部の機能に対し，チームメンバの理解が曖昧なままだったことが確認された．

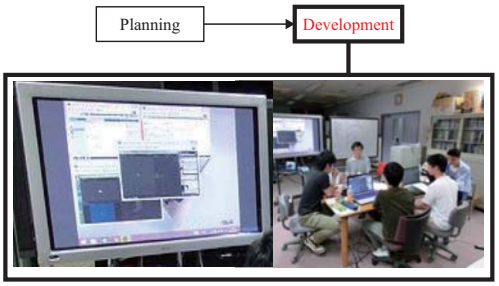

図 3 共有された作業画面

(8) **技術的な問題発覚**：開発過程において，バグの修正作業や実現の難しい機能の切り捨てが多く発覚した．

(9) **実装後の修正要求**：実装された機能をメンバが確認した際，「見栄えが良くない」とし，デザインに関して新たな要求を提示したものの，ハッカソン時間内にその要求が反映されることはなかった．

(10) **待機状態化**：あるメンバは，「予定の作業が終わったが，次に控えている作業を把握できておらず，何をすればいいか判断できなくなった」と述べた．

(11) **優先順位の誤り**：あるメンバは，成果物のコンセプトに直接関係のない機能の実装に長時間注力していた．

(12) **リソースの未使用・誤使用**：あるメンバは多くのテキストデータや画像素材などを用意したが，最終的なアプリケーションに反映されたものはごくわずかだった．また，別チームのメンバは，用意した効果音が「意図せぬ用途で使われた」と述べた．

4. HackathonMediator

予備実験の結果に基づき，提案システム HackathonMediator は「成果物イメージ共有を兼ねたプロトタイプ Ver.0 作成機能」および「作業進捗共有を兼ねた画面共有機能」を提供する．各機能の支援内容を以下に述べる．

4.1 成果物イメージ共有を兼ねたプロトタイプ Ver.0 作成機能

本機能は，主に企画終了段階にて「成果物の完成イメージを遅延なく明確化・共通化する」ことを目的とする．これは，企画段階において，「開発への焦り」から，成果物のデザインや実装内容の議論が十分に行われず，「成果物に抱くイメージの差異」が発生していると考えられ，これが開発過程での「技術的な問題発覚」や「優先順位の誤り」などの問題に影響をもたらしていると思われるためである．そのために，アプリケーションスケッチなどの作成工程を「成果物イメージに関する議論」として完結させるのではなく，「開発段階での初期作業」として導入可能にする．

本機能では Nulab 社の共同デザインツールである Cacoo[*2] を活用する．Cacoo では，多くの UI パーツを模した

[*2] https://cacoo.com/

図 4　Cacoo のステンシルテンプレート

画像が「ステンシルテンプレート」として提供されており（図 4），ドラッグ&ドロップ操作で手軽にアプリケーションの画面スケッチ（モックアップ）を作成できるほか，複数のユーザでの共同編集が可能である．本機能では，Cacoo で作成したモックアップを実際のソースコードに変換可能として，開発段階への移行をシームレス化することにより，「開発への焦り」の防止作用を期待する．

本機能と Cacoo を連携する上で，Cacoo で使用可能な専用シートを提供する（図 5）．シート上の “CONVERT” ボタンを押下すると，モックアップの内容が解析され，アプリケーションとして即実行可能なソースコード群（プロトタイプ Ver.0）が生成される．このシート上には，モックアップの作成を行う「デザイナの作業領域」と，ソースコード中で使用予定の変数やメソッドを記述する「プログラマの作業領域」が混在している．このシート上でチームの全員が共同作業を行うことにより，「成果物イメージの差異」の解消を期待する．ソースコード変換の際，「デザイナの作業領域」の外に設置されたテキストオブジェクトは，「注釈オブジェクト」として解釈され，テキストに埋め込まれた「タグ」の内容に基づき，プロトタイプ Ver.0 のソースコードに空のメソッド等が挿入される（図 6）．

本機能で主に使用するタグを以下に示す．

- **[outlet]**　クラス名とオブジェクト名を定義する．
  記述例）**[outlet]NSArray* myArray**
- **[action]**　メソッド名と引数・返り値を定義する．
  記述例）**[action]-(void) myFunction**
- **[comment]**　コメント文として，オブジェクトまたはメソッド宣言文の真上に挿入される．直前のタグが [outlet] だった場合はオブジェクトに対するコメント文となり，直前のタグが [action] の場合はメソッドに対するコメント文となる．
  記述例）**[comment]**
  〇〇して××した値を返すメソッドです．
  △△君実装よろしく

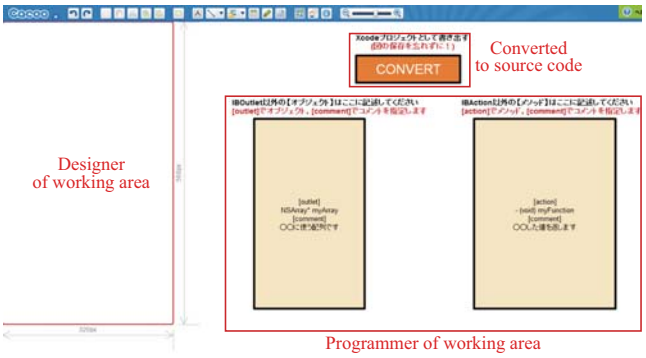

図 5　成果物イメージ共有を兼ねたプロトタイプ Ver.0 作成機能

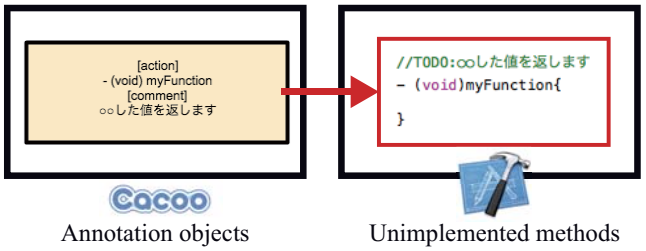

図 6　注釈オブジェクトのソースコード変換例

開発過程にて，挿入された空のメソッドの内部をプログラマが逐次実装していくことで，プロトタイプ Ver.0 を「プロトタイプ Ver.1」へと変化させていく．2015 年 2 月現在，本機能は iPhone アプリケーション開発環境である Xcode に対応している．

4.2　作業進捗共有を兼ねた画面共有機能

本機能は，主に開発過程にて「チーム内での作業進捗の共有を確実なものにする」ことを目的とする．これは，開発過程での作業に追い込まれることで「他メンバへの関心の希薄化」が生じ，「情報共有の頻度低下」や「共有された情報の確認不足」が発生し，「進捗状況の把握不足」や「優先順位の誤り」などの問題発覚の遅れに繋がっていると思われるためである．

本機能の概要を図 7 に示す．プログラマがアプリケーションのデバッグを開始した際や，デザイナやプランナがプレゼンテーション資料や画像ファイルを作成・編集した際，それらのアクションを検知し，該当メンバの作業画面を大型ディスプレイにミラーリングする（図 8）．これにより，「情報共有の頻度低下」の問題を解決する．またその際，効果音や通知メッセージの発行などにより，他メンバの作業を停止させ，共有画面の確認を促す（図 9）．なお，相手の作業を停止させる通知メッセージは，時としてチームに不利益をもたらす恐れがある．そのため通知メッセージには 3 段階のレベル設定を設けた（表 1）．画面共有時，各メンバは作業内容の重要度に合わせてこれらのレベルを設定する．これにより，「他メンバへの関心の希薄化」，「共有された情報の確認不足」の解消を図る．

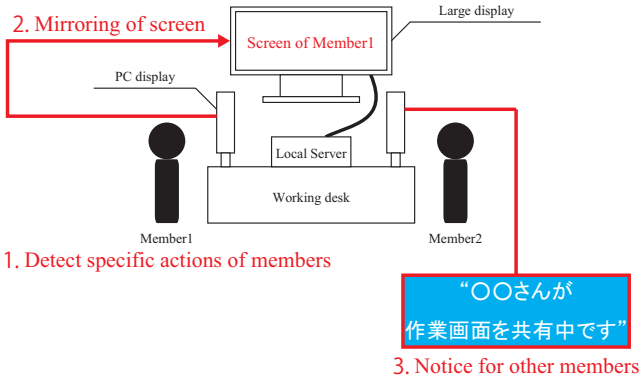

図 7　作業進捗共有を兼ねた画面共有機能

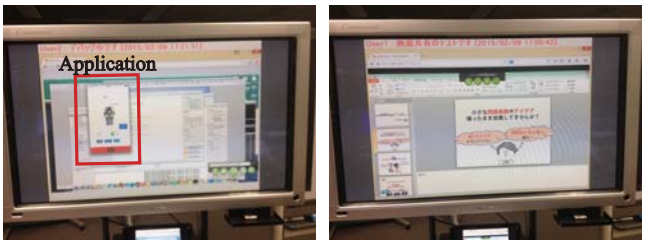

図 8　大型ディスプレイにミラーリングされた作業画面

図 9　他メンバに対し発行された通知メッセージ

表 1　通知メッセージのレベル

Level 0	画面のミラーリングが実行され 効果音が鳴るが，通知メッセージは発行されない
Level 1	画面をミラーリングした上で， 効果音と共に通知メッセージが発行される 表示されたメッセージはキー操作で削除できる
Level 2	画面をミラーリングした上で， 効果音と共に通知メッセージが発行される 共有者が画面共有を取り消すまで， 他ユーザは一切の作業が不可能になる

5.　実験

「成果物イメージ共有を兼ねたプロトタイプ Ver.0 作成機能」（以下，Ver.0 作成機能）および「作業進捗共有を兼ねた画面共有機能」（以下，画面共有機能）を実際のハッカソンに導入し，その効用を確認した．Ver.0 作成機能を導入するにあたり，参加者には「Xcode を活用した iOS アプリケーションの開発経験があること」を求めた．筆者らが所属する大学院内において条件に該当する参加者を確保することは難しかったため，本実験は 2015 年 1 月 30 日に九州産業大学情報科学部で開催された 10 時間ハッカソン「9389」にて実施した．このハッカソンでは，「学生生活に役立つ iPhone アプリケーションを作る」ことを共通のテーマとして設定している．また，ハッカソンの開始時には，ハッカソンの概要と本システムの仕様について参加者に説明を行っており，終了後にはアンケートを実施した．

5.1　参加者

- チーム A
  - (1) プログラマ A（チームリーダ，プレゼンテータ）
  - (2) プログラマ B
  - (3) プログラマ C（プレゼンテータ）
  - (4) デザイナ
- チーム B
  - (1) プログラマ A
  - (2) プログラマ B
  - (3) プログラマ C
  - (4) デザイナ（プレゼンテータ）

参加チームは 2 チームであり，チーム A，B 共にプログラマ 3 名，デザイナ 1 名の計 4 名で構成される．参加者の全員がプログラミング経験者であり，チーム A のプログラマ A とプログラマ C，チーム B のプログラマ A とプログラマ B は Xcode の活用歴が半年以上である．なお，参加者の過去のハッカソン経験については，全員が「経験したことがない」と答えている．また，チーム B においては明確なチームリーダが存在しなかった．

5.2　チーム A の観察

企画段階での議論にて，チーム A が作成したスケッチを図 10 に示す．チーム A はプレゼンテーションで強調するコンセプトやアプリケーションの機能に関して 1 時間程度の議論を行った．チーム A の企画内容は「エナジードリンクを飲むたびにアプリのボタンを押してポイントを溜め，一定のポイントが溜まる度にレベルが上がっていき，そのレベルをみんなで競うアプリ」だった．

5.2.1　モックアップの作成

企画立案終了時に，チーム A が作成した Cacoo のモックアップを図 11 に示す．チーム A は 1 つのモックアップを作成しており，注釈オブジェクト内で内部処理に関わる変数およびメソッドが複数定義されていた．

5.2.2　開発過程での行動

図 12 に画面共有履歴の一部を示す．画面共有機能の「デバッグの自動検知」により，Xcode での開発をメインとしていたプログラマ C の画面が高い頻度で共有されていた．次点で共有頻度が高かったのはプログラマ B の画面であり，主にプレゼンテーションの資料作成時の「ファイル作

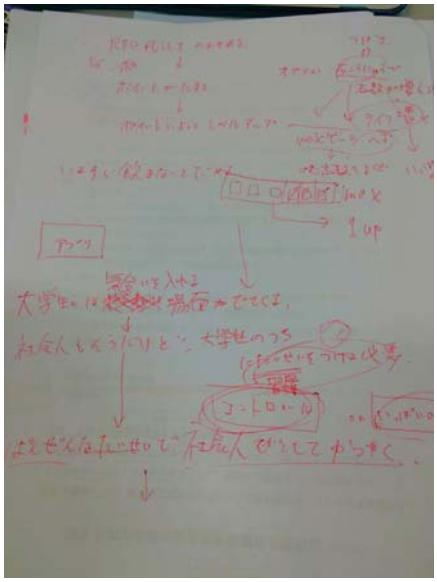

図 10　企画段階でチーム A が作成したスケッチ

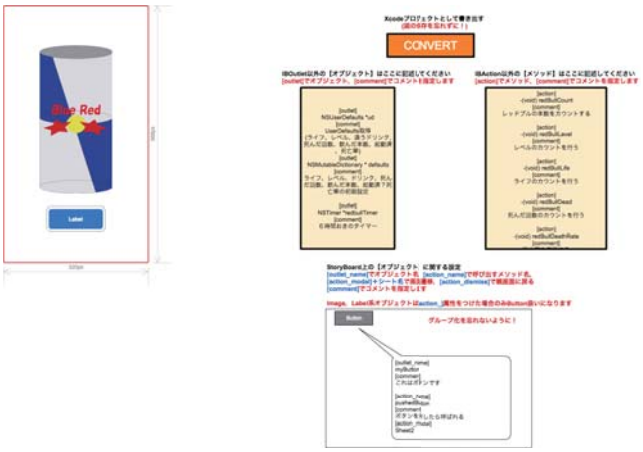

図 11　企画終了時にチーム A が作成した Cacoo モックアップ

図 12　画面共有履歴の一部（チーム A）

成・編集の自動検知」によるものであった．プログラマ C の画面共有時の共有レベルはほぼ常に 1 であり，デバッグ時にはチームメンバから「おー」などの反応を頻繁に受けていた．この反応を受けたプログラマ C は，現在のアプリケーションの状態を報告するなどの行動をとった．

5.2.3　完成したアプリケーション

チーム A が最終的に完成させたアプリケーションの状態を図 13 に示す．チーム A はプロトタイプ Ver.0 作成時に

First View (Default pattern)　First View (Another pattern)

図 13　完成したアプリケーション（チーム A）

```
//TODO:レッドブルの本数をカウントする
-(void) redBullCount{
    ud = [NSUserDefaults standardUserDefaults];
    NSInteger count_check = [ud integerForKey:@"KEY_COUNT"];
    count_check = count_check+1;
    _countLabel.text = [NSString stringWithFormat:@"%ld本",count_check];
    if(count_check%5==0 && dead_swich == NO){
        [self redBullLevel];
    }

    [ud setInteger:count_check forKey:@"KEY_COUNT"];
}
//TODO:レベルのカウントを行う
-(void) redBullLevel{
    ud = [NSUserDefaults standardUserDefaults];
    NSInteger level_check = [ud integerForKey:@"KEY_LEVEL"];
    level_check = level_check+1;
    [sound3 play];
    _levelLabel.text = [NSString stringWithFormat:@"Lv.%ld",level_check];
    [ud setInteger:level_check forKey:@"KEY_LEVEL"];
    [ud setInteger:3 forKey:@"KEY_LIFE"];
    _lifePic1.hidden = NO;
    _lifePic2.hidden = NO;
    _lifePic3.hidden = NO;
}
```

Annotation Object

```
[action]
-(void) redBullCount
[comment]
レッドブルの本数をカウントする

[action]
-(void) redBullLevel
[comment]
レベルのカウントを行う
```

図 14　完成したアプリケーションのソースコードの一部（チーム A）

アプリケーションの画面数を 1 つに設定しており，最終的に完成したアプリケーションも 1 つの画面で構成されたものだった．

チーム A が最終的に完成させたアプリケーションのソースコードの一部を図 14 に示す．チーム A は，Ver.0 作成機能のソースコード変換機能を使用し，モックアップから変換されたソースコードを用いて開発を行っていた．アプリケーションのソースコード中において，いくつかのオブジェクトとメソッドに関する宣言文およびコメント文が，ソースコード変換機能によって挿入されていることを確認できた．ソースコード変換機能によって挿入されたメソッドは 7 つであり，その内 6 つのメソッドの内容がメンバによって実装されていたことを確認した．スケッチ作成時に議論された機能はこれらのメソッド内で実装されていたほか，「別バージョンの操作画面を表示する機能」や「効果音を再生する機能」など機能拡張が見られた．

5.3　チーム B の観察

5.3.1　スケッチの作成

企画段階での議論にて，チーム B が作成したスケッチを図 15 に示す．チーム B はこのスケッチを記述しつつ，アプリケーションのコンセプト，機能，画面のデザイン，実装方法，役割分担に関して 2 時間程度の議論を行った．チーム B の企画内容は「大学講義の内容について受講者同士がチャットできるアプリ」だった．

5.3.2　モックアップの作成

企画終了時にて，チーム B が作成した Cacoo のモック

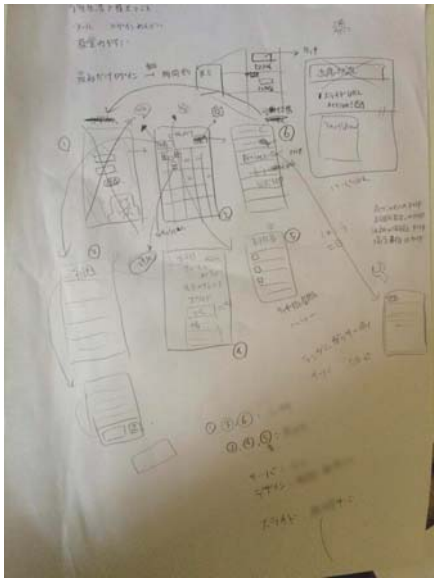

図 15　企画段階にチーム B が作成したスケッチ

図 16　企画終了時にチーム B が作成した Cacoo モックアップ

アップを図 16 に示す．チーム B は 2 つのモックアップを作成しているが，注釈オブジェクトに関してはテンプレートシート内に一切の記述が残されておらず，またソースコード変換機能の使用履歴も確認されなかった．

5.3.3　開発過程での行動

図 17 に画面共有履歴の一部を示す．画面共有機能の「デバッグの自動検知」により，プログラマ B の画面がほぼ常に共有される状態にあった．画面共有時において，プログラマ B の共有レベルは常に 0 であった．開発が終盤に近づくにつれ，チーム B からは「ヤバイ」といった内容の発言が増えるようになり，その後全てのプログラマがプログラマ B の席に集まり，実装の済んでいない機能を解決しようとする動きをとった．プログラマらのこの行動により，デザイナのみ対面側の席で 1 人作業する状態となった．その後「デバッグの自動検知」によってプログラマ B の作業画面が共有された際，画面を確認したデザイナが様子を伺いにプログラマ B の席に近づき，状況を確認しようとするなどの行動が見られた．

5.3.4　完成したアプリケーション

チーム B が最終的に完成させたアプリケーションの状態を図 18 に示す．チーム B は当初の企画においてアプリケーションの画面数を 6 つに設定していたが，最終的に完成したアプリケーションは 3 つの画面で構成されていた．

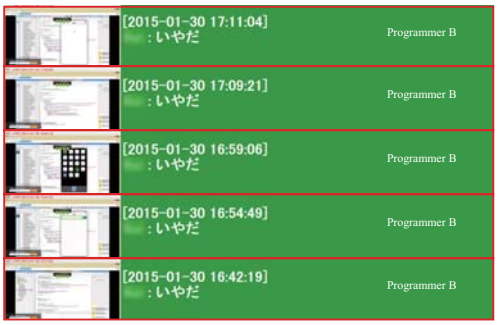

図 17　画面共有履歴の一部（チーム B）

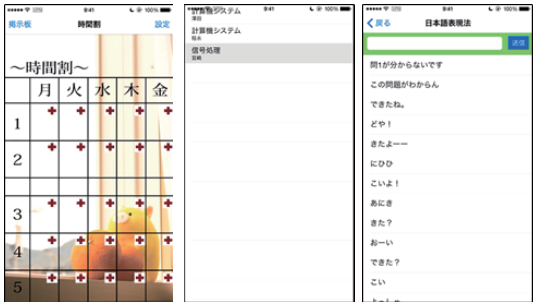

図 18　完成したアプリケーション（チーム B）

6.　考察

HackathonMediator の 2 つの機能について，実験結果からの考察を述べる．

6.1　「成果物イメージ共有を兼ねたプロトタイプ Ver.0 作成機能」に関する考察

実験において，チーム B はあくまでも Cacoo の提供している機能を活用したのみに留まっており，ソースコード変換機能を使用しなかったため，本機能を「活用した」といえるのはチーム A のみであった．チーム A の行動観察から，スケッチの作成段階で議論されていた機能が，最終的に完成したアプリケーションの中に反映されていることを確認した．このアプリケーションの根幹部分は，プロトタイプ Ver.0 内に記述されたメソッドの内容に基づいて実装されていたことがソースコードから確認できている．この根幹部分をベースとして，アプリケーションには「別バージョンの操作画面を表示する機能」や「効果音を再生する機能」などの機能拡張が施されていた．また，アンケートからは「最初にチームでソースコードを共有できたので，必要な部分だけ開発を行えた」，「最初に必要なメソッドなどを書き込んでおくことで，あとの開発での管理が行いやすかった」といった開発時に得られたメリットも述べられていた．以上から，プロトタイプ Ver.0 の作成過程での作業が，開発時における「優先順位の誤り」などを防止する

役目を果たしたのではないかと考える．本機能の有用性に関するアンケートでは，チーム A の全員が「有用だと感じた」と述べており，「この機能を使えば UI がどのような動作を担おうとしているのか分かりやすい」，「リアルタイムで他の人と一緒に作業できた」などの感想が得られた．このように，本機能が「成果物イメージの差異」の解消をもたらす可能性が見受けられた．

次に，チーム B が「本機能を活用しなかった理由」について考察する．チーム A の企画段階では，主に「コンセプト，プレゼンテーション，機能」についてのみが議論された．これに対し，チーム B は企画段階でのスケッチにて「コンセプト，機能，画面のデザイン，実装方法，役割分担」に至るまでが一通り議論されていた．その結果，チーム B の議論はチーム A に比べて 1 時間程度長引いている．その後チーム B は，モックアップ作成段階にて 2 つのモックアップを作成しチーム内で共有を図っていたが，この中にアプリケーションの TOP 画面となった「時間割画面」が存在していないなど，作られたモックアップは完全なものではなかった．アンケートにて「時間に追われた作業だったので活用する余裕が無かった」ことがメンバから述べられたことから，チーム B では「開発への焦り」に関連する問題が発生したと思われた．本来であれば，企画段階での「画面のデザイン，実装方法，役割分担」に関する議論は，本機能を活用する中で行われるべきであったと考えられ，チーム B がコンセプトや機能の議論を終えた時点で「続きは本機能を用いて議論すること」を指示すべきであったと考える．

6.2 「作業進捗共有を兼ねた画面共有機能」に関する考察

共有された画面を確認したメンバが意見や感想を述べたり，様子が気になり近くに駆けつけ，それを受けた共有者が説明を行う場面が双方のチームで確認された．被験者から「意見を募りやすかった」，「みんなどんなことをしてるのかわかった」などの感想が得られたことからも，本機能はチームに対し情報共有の機会をもたらす作用が存在すると考えられ，「他メンバへの関心の希薄化」の解消をきっかけとして「情報共有の頻度低下」や「共有された情報の把握不足」の問題を解消できる可能性が見受けられた．しかし，本機能が目指した「チーム内での作業進捗の共有を確実なものにする」ことに関しては，プライバシーなどの心理面の問題が今後の課題だと思われた．被験者に実施したアンケートにて，「必要がないときも画面が共有される」，「自動でミラーリングされるのは恥ずかしい」など，画面共有そのものに拒否感を抱く意見が述べられていた．この問題に関しては「公開可能な画面の範囲」などを設定可能にすることを今後検討していく．

7. まとめ

本稿では，ハッカソンチームの失敗要因について調査を行い，それらの問題を解決するためのハッカソン支援システム HackathonMediator を開発した．本システムを実際のハッカソンに導入し，参加メンバに対して実施した行動観察やアンケートなどから，「成果物イメージ共有を兼ねたプロトタイプ Ver.0 作成機能」と「作業進捗共有を兼ねた画面共有機能」によって，ハッカソンで生じる多くの問題に関する解決可能性が見いだせた．

今回実施した評価は，非常にわずかな被験者数に対して実施した行動観察とアンケートに基づくものであった．本システムの効用を定量的に明らかにする上で，さらなる被験者の協力が必要である．今後，さらにシステムを改良し，より多くのハッカソンチームに本システムを導入してもらうようにつとめたい．

謝辞 本研究は，科学研究費助成事業（課題番号 26280126）の支援を受けて実施されました．

参考文献

[1] Kwan, I., et al.: Visualizing a requirements-centred social network to maintain awareness within development teams, Proc. 1st Int'l. workshop on Requirements Engineering Visualization, p.7, 2006.
[2] Bashir, M. Salman, and M. Rizwan Jameel Qureshi: Hybrid Software Development Approach For Small To Medium Scale Projects: Rup, Xp & Scrum, Sci.Int.(Lahore), 24(4), 381-384, 2012.
[3] Shaw, A., et al.: Computer supported collective action, Interactions 21.2, pp.74-77, ACM, 2014.
[4] Ryan, S. and O' Connor, R. V.: Acquiring and sharing tacit knowledge in software development teams: An empirical study, Information and Software Technology, Vol.55, Issue 9, pp.1614-1624, 2013.
[5] Lamela, J. L. Z., et al.: Hacking sustainability: Broadening participation through green hackathons, In 4th Int'l. Symp. on End-User Development, 2013.
[6] 清水たくみ: オープンデータ活用によるアプリケーション開発:ハッカソン実態調査を通じて, 組織学会大会論文集, pp.38-43, 2013.
[7] Raatikainen, M., et al.: Industrial experiences of organizing a hackathon to assess a device-centric cloud ecosystem, Proc. Computer Software and Applications Conference (COMPSAC), pp.790-799, 2013.
[8] Musil, J., et al., Synthesized essence: what game jams teach about prototyping of new software products, Proc. 32nd Int'l. Conf. on Software Engineering, Vol.2, pp.183-186, 2010.