OSS 開発における保守対応の効率化のためのアウェアネス支援システム

伊原彰紀[†1]　山本瑞起[†1]
大平雅雄[†1]　松本健一[†1]

ミッションクリティカルなシステムで利用されているオープンソースソフトウェア (OSS) は，OSS 開発者によって不具合を迅速に修正する必要がある．しかしながら，OSS は言語や文化，習慣等が異なる世界中の不特定多数のボランティア集団によって開発を進められているため，OSS 管理者が保守作業の適任者を特定することが容易でない．そこで我々は，OSS の保守作業を依頼すべき開発者を効率的に決定できるようにするためにアウェアネス支援システム ACTION（Awareness Communication Tool for Interactive Open Negotiation）を開発した．ACTION は (a) 保守対象のソースコードの開発に関与した経験のある開発者, (b) 開発者の活動地域と開発者の活動時間，を OSS 管理者が直観的に把握できるよう支援することができる．本稿で，ACTION が開発者の活動地域や活動時間によって生じるコミュニケーションの遅延を軽減できることを確認した．

An Awareness Support System for Maintenance in OSS Development

Akinori Ihara,[†1] Mizuki Yamamoto,[†1] Masao Ohira[†1] and Ken-ichi Matsumoto[†1]

Open source software (OSS) has become to be used in mission-critical systems. In such the context, OSS must be reliable. When bugs in OSS are found, OSS managers need to ask developers to fix bugs as soon as possible they can. However, it is not easy for OSS managers to identify adequate developers for individual bugs, due to the differences of language, culture, and customs among geographically-distributed, volunteered OSS developers. In this paper, we construct a system called ACTION (Awareness Communication Tool for Interactive Open Negotiation), to help OSS manager identify adequate developers for bug modifications. ACTION provides visualized information on (1) developers who have modified source codes including bugs and (b) active area and active time where OSS developers live. As the results of our experiment, we have confirmed that ACTION could help managers to effectively communicate with developers.

1. はじめに

近年，Linux をはじめとするオープンソースソフトウェア (OSS) は，個人ユーザによる利用のみならず，基幹業務システムなど企業におけるミッションクリティカルな業務での利用が進んでいる．OSS の社会的重要性が高まるにつれて，たとえボランティアの開発者によって開発される OSS であっても，発見・報告される OSS の不具合に対する迅速な対応が強く求められている．特に，セキュリティの脆弱性に関連する不具合は，悪意のある他者からの攻撃を避けるためにも早急に対処される必要がある．

迅速な不具合修正を実現する方法の 1 つとして，不具合を混入した開発者を不具合修正担当者として割り当て保守作業を実施する方法が考えられる．ソースコードへ不具合を混入した当事者であると同時にソースコードの開発者であるため，当該ソースコードに対する理解は必然的に高く，素早くかつ的確に保守作業に取り掛かることができるためである．この方法に従い保守作業を行うために，プロジェクト管理者はまず開発プロジェクトに所属する全開発者の中から不具合を混入した開発者を特定し，迅速な保守作業を依頼する必要がある．しかしながら，OSS 開発においては一般に，保守作業の適任者を特定することは容易ではない．ソースコードは複数人の開発者によって共同開発されることが多いため，不具合がどのタイミングでどの開発者によって混入されたものなのかが明確でないからである．

次善策として当該ソースコードの開発に関与した経験のある開発者のいずれかに保守作業を依頼するという方法を採ることができるが，OSS 開発の参加形態の特徴により，保守を依頼された開発者が素早く作業に着手することを保証できないという問題が新たに生じる．OSS 開発の参加形態の特徴により生じる問題とは，(1) OSS 開発プロジェクトでは開発者は自由意思で開発に参加しており，依頼される作業を遂行する責務を一般には負わないため，依頼を断られた場合は何度も他の開発者に依頼する必要があること，また，(2) 開発者は世界各地に分散しており（すなわち，タイムゾーンが異なる），かつ，それぞれの生活

†1 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

サイクルの中で開発に参加している（例えば，就業後や休日にのみ OSS 開発に参加するなど）ため，緊急性の高い不具合の修正において必要になる開発者間の密なコミュニケーションが困難なことである．

上記の問題に対処するために OSS プロジェクト管理者は，OSS の開発や保守に関する情報交換のために用いられるメーリングリストに流れる情報や，作業成果物を蓄積・管理するためのバージョン管理システムや不具合情報を管理する不具合管理システムの履歴を常に追い続け，不具合が発生した際にどの開発者に保守を依頼すべきかを把握しておかなければならない．しかしながら，近年の OSS の大規模化・複雑化に伴い報告される不具合も年々増加してきており，それらすべての情報を把握し的確に指示を与えることは現実的に不可能な状況にある．

そこで我々は，OSS プロジェクト管理者が保守作業を依頼すべき開発者を効率的に決定できるようにするアウェアネス支援システム ACTION（Awareness Communication Tool for Interactive Open Negotiation）を構築した．ACTION は，(a) 保守対象のソースコードの開発に関与した経験のある開発者, 及び (b) 開発者の活動地域と開発者の活動時間，をプロジェクト管理者が直観的に把握できるよう支援するシステムである．

本稿では，ACTION の評価実験を通じて OSS 開発者間の活動時差によって生じる「コミュニケーションの遅延」を改善できるか評価する．

2. OSS 開発におけるアウェアネス支援

本章では，OSS 開発におけるアウェアネス情報の重要性について述べた後，既存のアウェアネス支援システムと，本研究のアウェアネス支援システムとの違いを述べる．

2.1 OSS 開発におけるアウェアネス情報

一般的にアウェアネスとは複数人による共同作業を行う際の状況情報への「気づき」を指す．状況情報とは共同作業メンバーの状況（何をしているのか，どこにいるのか，など）や，メンバーの作業によって得られた成果物についての状況（成果物の進捗など）に関する情報を指す．複数人で行うソフトウェア開発プロジェクトの共同作業では，開発者は開発者に関する状況，成果物の進捗状況，などを把握することが必要不可欠である[4]．開発に関するコミュニケーションを行う上で，共同作業を一緒に行うメンバーがどこで活動しているのか，どのような状況にあるのか，などアウェアネスを確保することが重要である．

表 1 では，同一場所で活動する企業のソフトウェア開発，遠隔場所で活動するオフショア開発，OSS 開発の 3 種類のソフトウェア開発を地理的空間と時間の観点から分類した．同

表 1 ソフトウェア開発の分類
Table 1 Classification of software development

地理的空間＼時間	リアルタイム型	非リアルタイム型
対面型	企業のソフトウェア開発	該当なし
分散型	オフショア開発	OSS 開発

一場所で活動する企業のソフトウェア開発はコミュニケーションを行うとき，相手の状況情報を理解した上でコミュニケーションを行うことが可能である．一方，オフショア開発は，地理的に異なる場所で活動していても，大抵の開発グループはお互いがどこで働いているか把握しているため，リアルタイムのコミュニケーションが可能である．ただし，地理的に異なる場所で活動しているため，お互いに何をしているかという状況情報を共有することは困難である．OSS 開発プロジェクトでは，世界中の不特定多数の開発者がプロジェクトに参加しているため，同一時間に活動することが困難であり，お互いの情報状況を把握していない開発者間が円滑なコミュニケーションを図ることは他のソフトウェア開発に比べて最も困難であると考えられる．

Hervsleb，Grinter らは，複数の開発拠点で実施されるソフトウェア開発では，開発者間の対面コミュニケーションの機会が減り，開発者のアウェアネス確保が困難であることを報告している．また，アウェアネス情報の不足は開発に関する「コミュニケーションの遅延」を生じさせ，ソフトウェアの不具合修正に遅延をもたらしていることを示唆し，アウェアネス情報の重要性を指摘している[5]．

そこで，OSS 開発者はアウェアネスの不足を補うために，メンバーが過去に行った作業結果や作業履歴情報を開発者が共有することで，現在進行中の作業を補助している．例えば，開発者はバージョン管理システムや不具合管理システムの過去の作業履歴を参照することで，開発者によって生じた「成果物に関するアウェアネス情報の確保」を実現している．

しかしながら，近年の OSS の大規模化・複雑化に伴い報告される不具合も年々増加してきており，それら全ての情報を把握し的確に指示を与えることは現実的に不可能な状況にある．OSS 開発者にインタビューした結果によれば，開発者は常にメーリングリストやバージョン管理システムの履歴を追い続けなければアウェアネス情報を得ることは困難である[4]．そのため，OSS 開発者が保守作業を円滑に行うためには，OSS 開発プロジェクトに参加する開発者のアウェアネス情報を把握することが必要である．

2.2 既存のアウェアネス支援システム

これまで，ソフトウェア開発を支援するために，多くのアウェアネス支援システムの研究が行われている[1)2)3)8)]．ソフトウェア開発におけるアウェアネス支援システムは，開発者，マネージャー，テスターなど，誰を対象とするか，また，どんな情報を提示するかによってシステムの要件は大きく異なる[9)]．本稿では，OSS 開発者に過去の開発経験と開発者の活動地域・活動時間のアウェアネス情報を提供するシステムを開発する．

既存のアウェアネス支援システムには，モジュールの依存関係などを可視化し，開発を支援する Tukan[8)]，ソースコードとそのソースコードを変更した開発者の関係を可視化し，開発を支援する Ariadne[1)]，開発者がソースコードを変更した時間や，変更されたソースコードはソースコード中のどこに存在するのかを可視化し，開発を支援する Augur[2)]，ソフトウェア中のモジュール間の関係を可視化し，開発を支援する SoftChange[3)] などがある．

Ariadne は，開発者とソースコードの関係を可視化，支援しているため，どの共同開発者がどのような専門性を有しているかを把握することができる．そのため，Ariadne は「成果物に関するアウェアネスの確保」を実現できると考えられる．しかし，開発者間の活動地域の違いによる時差や生活サイクルの違いによる活動時差が考慮がされていないため,「コミュニケーションの遅延」を改善することはできない．

Augur は，どの共同開発者が，どのソースコードをいつ変更しているのかという情報を提示しているため,「成果物に関するアウェアネスの確保」を実現できると共に，活動時差によって生じる「コミュニケーションの遅延」を改善できると考えられる．しかし，開発者毎にタイムゾーンが異なるということが考慮されていないことから，時差によって生じる「コミュニケーションの遅延」を改善できない．

しかし，これらのアウェアネス支援システムを利用したとしても，OSS 開発における「成果物に関するアウェアネスの確保」の実現,「コミュニケーションの遅延」の改善が共にできるわけではない．そこで本稿では，OSS プロジェクト管理者が保守作業を依頼すべき開発者を効率的に決定できるようにするため,「成果物に関するアウェアネスの確保」を実現し，時差と活動時差によって生じる「コミュニケーションの遅延」を改善できるアウェアネス支援システムを構築する．

3. OSS の保守作業の効率化に向けたアウェアネス支援システム要件

本章では，OSS 開発における「成果物に関するアウェアネスの確保」,「コミュニケーションの遅延」を改善するためのアウェアネス支援システムの要件を述べる．

3.1 成果物に関するアウェアネスの確保を実現する方法

R1: モジュール担当者を特定する

開発者が各モジュールの過去に担当した開発者を把握できていないため，不具合修正を行う担当者の変更が何度も繰り返され，不具合修正が長期化していることが示唆されている[6)]．OSS 開発では，非対面のコミュニケーションによって情報共有が行われるため，開発者は常に，メーリングリストに流れるメールやバージョン管理システムの履歴を追い続けなければ，各モジュールの担当者を特定できない[4)]．従って，この手間を小さくし，各モジュール担当者を容易に特定することができる支援が必要である．なお，開発者の流動性が激しい OSS 開発では，モジュール担当者がプロジェクトから離脱するという事態もあることから，過去から現在までのモジュール担当者を把握できる支援を行う必要がある．

R2: モジュール担当者の開発履歴を共有する

ソフトウェアの大規模化，複雑化に伴い，開発者が増加することで，モジュール担当者の特定や，モジュール担当者の専門性を把握することが困難になり，不具合修正を行う担当者の割り当てが何度も繰り返され，不具合修正が長期化する要因になりうる．従って，モジュール担当者の開発履歴を把握するための支援を行う必要がある．

3.2 コミュニケーションの遅延を改善する方法

R3: 時差の影響を軽減する

これまでの研究で活動地域の異なる開発者間に時差があるため，開発者同士のコミュニケーションに遅延が生じ，不具合修正にかかる時間が長期化することが指摘されている．開発者が世界各国に点在し，開発される OSS では，時差の影響を根本的に解消することは不可能である．従って，開発者が共同開発者のタイムゾーンに合わした時間でコミュニケーションが出来る支援を行う必要がある．

R4: 活動時差の影響を軽減する

開発者間に時差が無い場合でも，開発者の生活サイクルの違いによる活動時差があるため，開発者同士のコミュニケーションに遅延が生じていることが報告されている[7)]．このことから，活動時差も時差同様，不具合修正にかかる時間を長期化させる問題であると考えられる．開発者が自主的に参加協力し，開発される OSS では，同一タイムゾーンであっても活動時差を無視することはできない．従って，開発者は共同開発者が開発に従事している時間（以降，活動時間と呼ぶ）と同じ時間にコミュニケーションが出来る支援を行う必要がある．

4. 提案システム：ACTION (Awareness Communication Tool For Interactive Open Negotiation)

本章では，前章の要件をもとに OSS 開発における保守対応の効率化のためのアウェアネス支援システム ACTION について述べる．

4.1 システムの概要

ACTION は，モジュール毎の開発担当者，開発者のタイムゾーン，開発者の活動時間を提示する．加えて，開発者の専門性が把握できるように過去に行った活動履歴（ソースコードの変更履歴）をリスト形式で開発者毎に提示する．OSS プロジェクト管理者は，提示された情報を基に，任意のモジュールに関して詳しい開発者を決定し，その開発者のタイムゾーンや活動時間に合わせて不具合修正の依頼を行うことができる．

図 1 に実装したシステム ACTION の表示画面を示す．ACTION はローカルで動くシステムである．ユーザが，Folder/File Tree View から任意のモジュールを選択することで，モジュールに変更を加えた「開発者が属するタイムゾーン」と「開発担当者の履歴」,「開発者の活動時間」,「開発者の全開発履歴」で構成される情報を，それぞれ Module History View, Active Time View，Commit History View に表示する．

4.2 システムの構成要素

ACTION の構成要素を，図 2 に示す．ACTION はユーザがモジュールを選択する Folder/File Tree View，可視化情報が表示される Module History View，Active Time View，Commit History View，4 つの View によってユーザインタフェースが構築されている．ユーザが任意のモジュールを選択した際，データベース操作部にイベントが送られ，データベースに検索キーが送られる．データベースからの取得データは，データベース操作部から，データ可視化部に送られる．データ可視化部は，グラフの出力命令をユーザインタフェースに送り，ユーザに可視化情報を提示する．

以下の説では，図 2 に示すの各モジュールについて述べる．

4.2.1 データベース

データベースは 2 つのテーブルから構成されている．

- モジュール毎のコミットログをまとめたテーブル

  図 3 に，バージョン管理システムからモジュール毎のコミットログ (変更を行った開発者，変更時刻，変更を行ったモジュール) をまとめたテーブル作成までの流れを示す．バージョン管理システムに，コミットログが保存されていることを利用して，モジュー

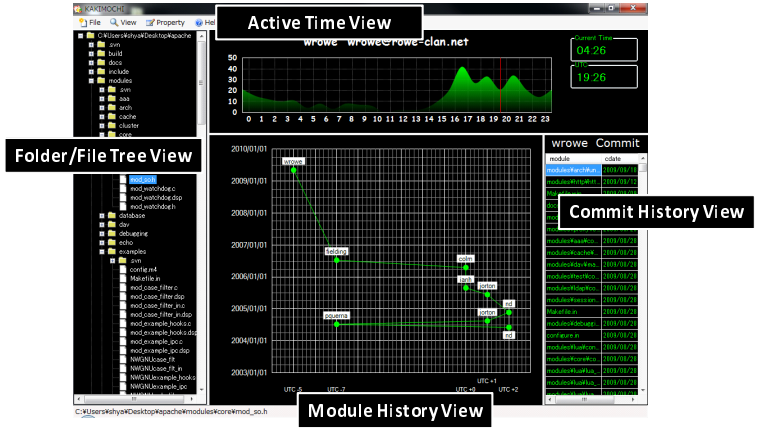


図 1 ACTION の全体図
Fig. 1 Overview of ACTION

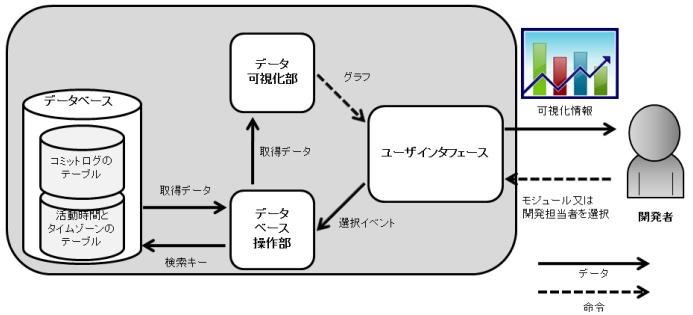
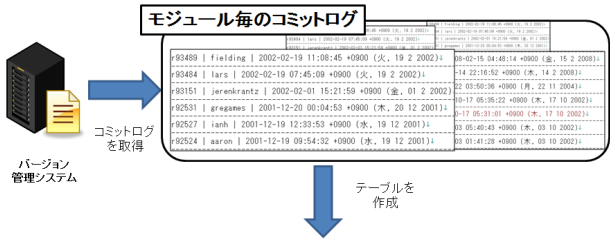
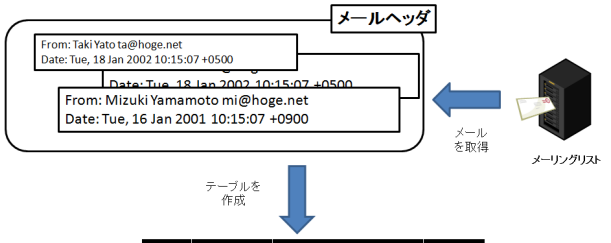


図 2 システムの構成要素
Fig. 2 System components



開発者	モジュール	日付	リビジョン番号
Mizuki	http.c	2008-02-15	r627857
tama	http.c	2002 -02-14	r627733
.	.	.	.
.	.	.	.
.	.	.	.
Mizuki	mod.c	2002-02-19	r93489
lars	mod.c	2002-2-19	r93484
.	.	.	.
.	.	.	.
.	.	.	.

図 3 コミットログをまとめたテーブルの作成
Fig. 3 Making a table for commit logs



開発者	メールアドレス	時刻			タイムゾーン (UTC)
		0-1	·····	23-0	
Mizuki	Miz@foo.pc	5	···	7	+5
jim	jim@hoge.jp	1	···	20	+6
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
fielding	mod.c	20	···	10	-1
lars	mod.c	5	···	2	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

図 4 メールヘッダ中のタイムゾーンと送返信時刻情報をまとめたテーブルの作成
Fig. 4 Making a table for time zone and posted time in e-mail headers

ル毎のコミットログを，開発者名，変更モジュール名，時間，リビジョン番号のフィールドにわけ，テーブルに格納することでモジュール毎のコミットログをまとめたテーブルを作成する．

- 開発者毎のタイムゾーンと，送返信時刻をまとめたテーブル
  図 4 に，開発者毎のタイムゾーンと，送返信時刻をまとめたテーブル作成までの流れを示す．メーリングリストに投稿されたメール毎に，開発者名，メールアドレス，タイムゾーンを取得する．また，メールの送信時刻をカウントし，開発者名，メールアドレス，活動時間，タイムゾーンのフィールドを持つテーブルに格納し，開発者毎のタイムゾーンと，送返信時刻をまとめたテーブルを作成する．

4.2.2 データ可視化部

Microsoft が提供している.NET Framework 対応のコントロールである Microsoft Chart Controls を使用し，データ操作部から受け取ったデータをグラフにする．

4.2.3 データベース操作部

.NET アプリケーションからデータベースに接続するために ADO.NET と呼ばれる.NET Framework のアクセス機能を使用し，検索キーやデータの受け取りを行っている．

4.2.4 ユーザインタフェース

- Folder/File Tree View
  図 1 の Folder/File Tree View には，ツリー構造化されたフォルダ・ファイルの一覧を示す．ユーザが任意のフォルダに含まれるファイルを探すことが出来るように，ツリー構造化し，提示している．ファイルをクリックすることで，イベントが起こり，ファイルを検索キーとしたクエリがデータベースに送られる．
- Module History View
  図 1 の Module History View には，ファイルの変更を行った開発者，変更時刻をグラフに示す．横軸はタイムゾーン，縦軸は日付である．1 プロットは開発者 1 名に該当する．プロット間を結ぶ線分は，開発者担当者の前後関係を明確にすると共に，線の長さで，モジュールの変更の頻度や時差の大小を提示している．例えば，横軸方向に線分が長い場合は，開発者間で時差が大きいことを意味し，縦軸方向に線分が長いときは，モジュールの変更の頻度が少ないことを意味している．
- Active Time View
  図 1 の Active Time View には，開発者の活動頻度を表すヒストグラムを示す．Active Time View は Module History View 内のプロットである開発者がクリックされた際，

当該開発者のメールの送信数をヒストグラムで提示する．横軸は UTC ± 0 に修正された時刻，縦軸はメールの送信数であり，グラフ上部には開発名とメールアドレスを表示する．ヒストグラム内に引かれている赤線はローカル時刻を UTC ± 0 時刻に修正したときのタイムバーである．また，ローカル時刻と UTC 時刻をヒストグラム右側に表示する．現在 Active Time View に示されている活動時間を読み取ると，開発者 wrowe は UTC ± 0 時刻 14 時から翌 4 時までメールの送返信を行っている．最も多くのメールの送返信を行っている時間帯は 16 時〜22 時付近であることが分かる．

- Commit History View
  図 1 の Commit History View には，開発者の開発履歴を示す．Commit History View では，Module Histoyr View 内のプロットである開発者がクリックされた際，当該開発者の開発履歴をデータベースから呼び出し，リスト化して表示する．

4.3 特徴・効果

本節では，システムの各 View とシステム要件を，対応させて列挙すると共に効果を述べる．

(1) Module History View (R3，R1)
Module History View では，過去のモジュール担当者を提示すると共に，その担当者のタイムゾーンを提示する．開発者は，コミュニケーションを取りたいモジュール担当者のタイムゾーンに合わせてメッセージを送ることが可能であるため，時差によって生じるコミュニケーションの遅延を軽減出来る．また，開発メンバーは，過去のモジュール担当者を把握できるため，開発者の流動性が激しい OSS 開発であっても，短時間で不具合修正が出来る担当者を選出することが可能である．

(2) Active Time View (R4)
Active Time View では，モジュール担当者のメールの送信時刻が偏っている時刻を提示する．開発者は，タイムゾーンの時間に合わせただけでは，遅延無しにコミュニケーションが出来なかったモジュール担当者と，遅延無しにコミュニケーションすることが可能である．

(3) Commit Hisotry View (R2)
Commit Hisotry View では，モジュールの担当者の開発履歴を提示する．複数のモジュールにまたがる不具合の場合であっても，モジュール担当者の開発履歴を確認することで，不具合が混入しているかどうか確認すべきモジュールを把握することが可能である．

5. 評価実験

5.1 実験概要

本実験は，提案システム ACTION を利用することで，開発担当者，開発担当者の活動時間を把握し,「コミュニケーションの遅延」が軽減できることを確認する目的で行う．実験では，複数の WEB ページの開発を想定し，WEB ページ毎に開発担当者がいるものとする．被験者は WEB ページの修正者として各 WEB ページを修正し，各 WEB ページの開発担当者から完成の確認を受けるというタスクを，ACTION 利用時と非利用時の計 2 セットを行なった．開発担当者は，実験中の任意の期間だけ対応できるもののとし，ACTION を利用しなければ，開発担当者の活動時間を把握することが出来ないものとした．被験者は奈良先端科学技術大学院大学の学生 10 名である．

5.2 実験のタスク

被験者は，テーブルレイアウトで作成された奈良先端科学技術大学院大学 情報科学研究科 ソフトウェア工学講座の WEB ページを，HTML と CSS に切り分け，同様な WEB ページを作製し，ページの開発担当者から，ページが完成しているかどうかの確認をもらうタスクを，ACTION 利用時と非利用時で計 2 セット行う．WEB ページで使われている画像ファイルやメニューなどの概観は予め用意しており，被験者は，ページの内容部分の HTML・CSS を作成する．なお，1 セット目と 2 セット目に作成する WEB ページは異なる．以下に，本実験で 1 セット目と 2 セット目の対象とした WEB ページを記す．

1 セット目の対象ページ

- 研究テーマ[⋆1]
- 設備[⋆2]
- 公開ソフトウェア[⋆3]
- アクセス・お問合せ[⋆4]

2 セット目の対象ページ

- ソフトウェア開発の計画・可視化[⋆5]

⋆1 http://se.naist.jp/html/research.html
⋆2 http://se.naist.jp/html/equipment.html
⋆3 http://se.naist.jp/html/software.html
⋆4 http://se.naist.jp/html/access.html
⋆5 http://se.naist.jp/html/analysis/index.html

- ヒューマン・コンピュータ・インタラクション[⋆1]
- ソーシャルネットワーク分析[⋆2]
- ソフトウェアレビュー[⋆3]

5.3 実験設定

実験を行う上で，被験者，ページの開発担当者，共通事項を設定した．以下でそれぞれについての設定を述べる．

被験者

対象ページを HTML と CSS に切り分け完成させる．HTML・CSS の編集は，全被験者同じエディタを利用する．また，実験中はインターネットを用いて HTML・CSS のタグについて調べることができる．

ページの開発担当者

活動時間のある実験中の限られた期間だけしか対応できない．ページが完成したかどうかの判断は，W3C Markup Validation Service[10] を用いて行い，エラーが無い場合は完成の連絡を行う．ページにエラーが含まれていた場合は，エラー修正指示を行う．なお，本実験ではページの開発担当者は実験者が担った．

共通事項

被験者が WEB ページを完成させたときの連絡，開発担当者からの完成確認の連絡は全てメーリングリストにメールを送信することで行われる．また，対象ページ毎に開発担当者がいるものとして，開発担当者毎に活動時間を定めた．

ACTION の利用，非利用の順序については，学習効果を考慮するために被験者がタスクを 2 回実施し，表 2 に示すように，ACTION を 1 セット目に利用しないグループ α と 1 セット目に利用するグループ β に分けた．また，被験者の HTML ・CSS の経験年数を用いて，スキルの差が生じないようにグループ α とグループ β に分けた．

5.4 実験の手順

(1) 実験の説明・準備：
実験についての説明を行う．なお，ACTION 利用時には，ACTION の使用方法を説明する．

(2) タスクの実施：
1 セット目または 2 セット目の対象ページを説明し，編集ファイルの所在を知らせる．また，メーリングリストにメールが送信できるかを確認して貰う．その後，対象ページを任意の順番で作成してもらう．タスクは 2 時間 1 セットで ACTION 利用時，非利用時で計 2 セット行ってもらう．

(3) インタビュー：
全タスク終了後に，どのよう順番で対象ページを作成したか，順番の理由などの質問に答えてもらう．

⋆1 http://se.naist.jp/html/HCI/index.html
⋆2 http://se.naist.jp/html/SNA/index.html
⋆3 http://se.naist.jp/html/review/index.html

表 2 実験条件
Table 2 Experiment condition

グループ	ACTION の利用	
	1 セット目	2 セット目
α	×	○
β	○	×

5.5 実験結果

本結果は，被験者が，対象ページを完成させたかではなく，ACTION を利用することで，各ページの開発担当者の活動時間を把握し，コミュニケーションの可否に着目したどうかについて分析した結果を述べる．

図 5 に 1 セット目の実験結果を示す．また，図 6 に 2 セット目の実験結果を示す．図 5, 図 6 には上から (a) 凡例，(b) ページの開発担当者の活動時間，そして，(c)，(d) は実験結果であるグループ α，グループ β の被験者が対象ページ作成に従事した活動時間を示す．各行は 1 人の被験者の活動内容を表している．例えば，図 5-(c) の被験者 A であれば，実験中に研究テーマのページのみ編集を行ったことが分かる．以下の小節では，1 セット目に ACTION 非利用から始めたグループ α と，1 セット目に ACTION 利用から始めたグループ β の結果を 2 つの観点（ページ担当者とコミュニケーションできた被験者数，対象ページへの時間配分）に分けて述べる．

5.5.1 ページ担当者とコミュニケーションできた被験者数

ACTION 非利用から始めたグループ α の 1 セット目 (図 5-(c)) は，ページの開発担当者とコミュニケーションできた被験者は 5 人中 0 人であったが，ACTION 利用の 2 セット目 (図 6-(c)) は，ページの開発担当者とコミュニケーションできた被験者は 5 人中 3 人となり，ACTION 非利用と比べて増えた．

ACTION 利用から始めたグループ β の 1 セット目 (図 5-(d)) は，ページの開発担当者

とコミュニケーションできた被験者は 5 人中 3 人であった．同様に，ACTION 非利用の 2 セット目 (図 6-(d)) でも，ページの開発担当者とコミュニケーションできた被験者は 5 人中 3 人であり変化は無かった．

5.5.2 対象ページへの時間配分

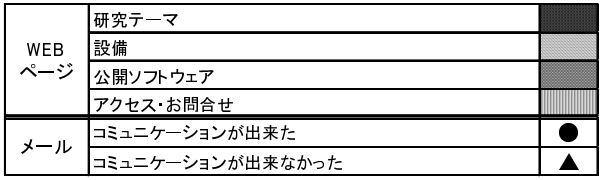
ACTION 非利用から始めたグループ α の 1 セット目 (図 5-(c)) は，被験者 D を除いて，被験者は 1 つの対象ページ作成に全時間を費やしている．ACTION 利用の 2 セット目 (図 6-(c)) は，被験者 B,C が非利用と変わらず，1 つの対象ページ作成に全時間を費やしているが，被験者 A,E は 1 セット目と異なり，複数の対象ページ作成に時間を費やしている．特に被験者 E の時間配分は，ページの開発担当者の活動時間 (図 6-(a)) に沿って時間を配分している事が分かる．例えば，ソフトウェア開発の計測・可視化のページ作成を開始 40 分までに終わらせ，次のページの開発担当者が活動するまでにページの作成を終わらせていることなどである．これは，被験者 E のインタビューの結果と一致した．

ACTION 利用から始めたグループ β の 1 セット目 (図 5-(d)) は，被験者 G を除いて，被験者はページの開発担当者の活動時間 (図 5-(b)) を考慮をした時間配分を行っている事が分かる．例えば，被験者 F は，開始 30 分で研究テーマのページの作成を止め，ページの開発担当者の活動時間がある設備のページの作成に取り掛かっていることなどである．ACTION 非利用の 2 セット目 (図 6-(d)) は, 利用時と比べて，被験者は 1 つの対象ページにかける時間が長くなっていることがわかる．

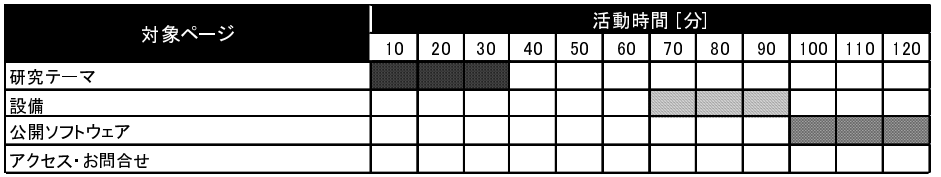
6. 考　　察

6.1 提案システム：ACTION

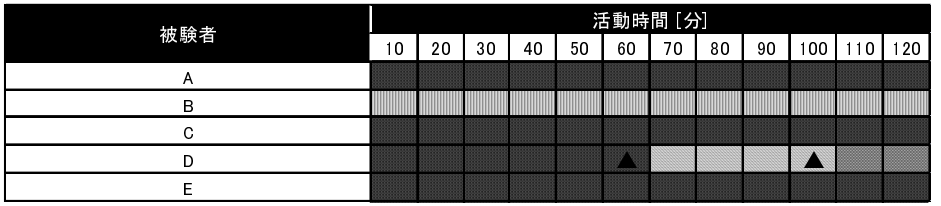
5.5 節より ACTION 非利用から始めたグループ α と ACTION 利用から始めたグループ β を比較すると，非利用から始めたグループ α は ACTION を利用することで，ページの開発担当者とコミュニケーションをすることができた被験者が増えたが，利用から始めたグループ β は，ページの開発担当者の活動時間に沿ってページの作成を行っているにも関わらず，ACTION 利用時，ACTION 非利用時，どちらもページの開発担当者とコミュニケーションをすることができた被験者数と変わらなかった．このような結果が得られたのは，2 つの理由が考えられる．1 つ目は，1 セット目で被験者が HTML，CSS を学習したために，2 セット目のページ作成の速度が向上し，ページの開発担当者の活動時間前にメールを送信することが出来たこと，2 つ目は，ソフトウェア開発の計測・可視化ページの開発担当者の活動時間が実験開始時刻から 30 分後までに加えて，実験開始から 60 分後からから 90 分後



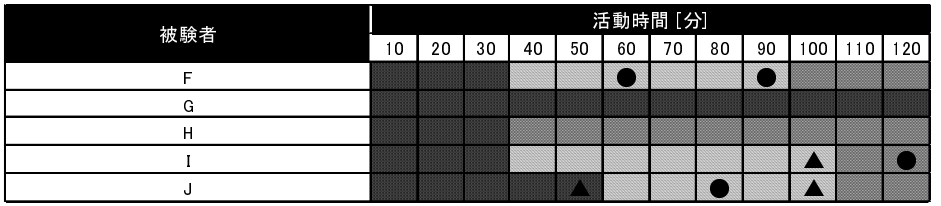
(a) 凡例



(b) ページの開発担当者の活動時間

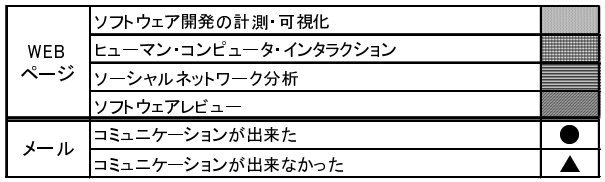


(c) 条件A：ACTION非利用時における被験者の活動時間

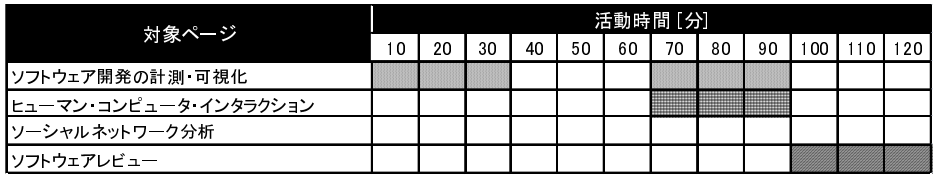


(d) 条件B：ACTION利用時における被験者の活動時間

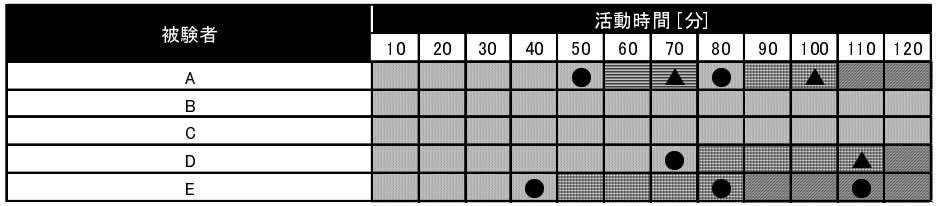
図 5　1 セット目：被験者の開発活動
Fig. 5　The first set: Subjects' development



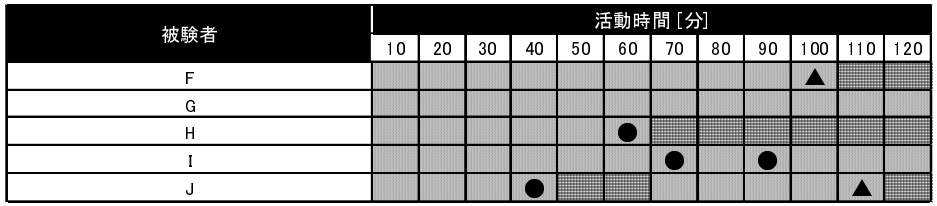
(a) 凡例



(b) ページの開発担当者の活動時間



(c) 条件A：ACTION利用時における被験者の活動時間



(d) 条件B：ACTION非利用時における被験者の活動時間

図 6 2 セット目：被験者の開発活動
Fig. 6 The second set: Subjects' development

の 2 回活動していること．この 2 つがページの開発担当者とコミュニケーションをすることができた被験者が増えた原因と考えられる．

学習による向上が無い 1 セット目の図 5 では，ACTION 非利用のグループ α に比べて，ACTION 利用のグループ β の方が，ページの開発担当者とコミュニケーションをすることができた被験者が多いことが分かる．この結果は，ACTION 利用のグループ β の方が，ページの開発担当者の活動時間 (図 5-(b)) を考慮して時間配分を行ったためであると考える．

学習による向上がある 2 セット目の 6 では，ACTION 利用のグループ α と ACTION 非利用のグループ β で，ページ担当者とコミュニケーションできた被験者数は変わらない．しかし，ACTION 利用のグループ α には，開発担当者がいる全ページで，コミュニケーションを行った被験者がいる．この結果は，ページ担当者の活動時間を考慮した時間配分で行ったためであると考えられる．

以上のことから，開発者間に活動時差がある OSS 開発であっても，ACTION を利用することで，開発者は共同開発者の活動時間に合わせた時間で開発，コミュケーションを行うことが可能と言える．従って，活動時差によって生じる「コミュニケーションの遅延」を軽減できると考える．

6.2 本論文の制約

被験者が，ACTION 非利用時に比べて利用時の方が，共同開発者の活動時間を把握し，開発，コミュニケーションをすることができたことから，活動時差によって生じる「コミュニケーションの遅延」を改善できたといえる．しかし，本システム ACTION が時差の影響を軽減するための支援できているかどうかについては，本論文では評価を行っていないため，今後評価する必要がある．

7. おわりに

本稿では，OSS 開発における保守対応の効率化に向けて「成果物に関するアウェアネスの確保」,「コミュニケーションの遅延」の要件を満たすためのにコミュニケーション効率化のためのアウェアネス支援システム「ACTION」を構築した．ACTION は，モジュール毎の開発担当者，開発者のタイムゾーン，開発者の活動時間を提示する．さらに，開発者の専門性が把握できるように過去に行った変更履歴を開発者毎にリスト形式で提示する．そのため，ACTION を利用する開発者は，提示された情報を基に，任意のモジュールに関して詳しい開発者や，コミュニケーションが遅延することなく開発を進めることのできるタイムゾーン，を把握しながら開発することができる．例えば．不具合修正を割り当てる開発者が

ACTION を利用することで,「成果物に関するアウェアネスの確保」を実現できるため，不具合修正の割り当てを適切に行うことが可能になると見込まれる．そのため，不具合修正を行う担当者の割り当てが何度も繰り返され，不具合修正が長期化している現状を改善できると考えられる．

本稿では，ACTION が活動時差によって生じる「コミュニケーションの遅延」を改善できるか評価した．ACTION 非利用時は開発担当者とコミュニケーションが出来なかったが，利用時は，コミュニケーションが取れるようになった．よって，ACTION を利用することで，活動時差によって生じる「コミュニケーションの遅延」を改善できることが分かった．

今後は,「成果物に関するアウェアネスの確保」に対する評価を行い，ACTION を用いることで保守作業の適任者を特定することが可能かどうかを判断した上で，ACTION がインターネット上から，開発者のタイムゾーンやメールの送返信数などを自動取得し，可視化情報を開発者に提示することによって不具合修正活動が円滑に行われるように支援したい．

謝辞　本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．また，本研究の一部は，文部科学省科学研究補助費（若手 B：課題番号 22700033）による助成を受けた．

参 考 文 献

1) de Souza, C.R., Quirk, S., Trainer, E. and Redmiles, D.F.: Supporting collaborative software development through the visualization of socio-technical dependencies, *Proceedings of the 2007 international ACM conference on supporting group work (GROUP2007)*, ACM, pp.147–156 (2007).
2) Froehlich, J. and Dourish, P.: Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams, *Proceedings of the 26th international conference on software engineering (ICSE2004)*, IEEE Computer Society, pp.387–396 (2004).
3) German, D.M., Hindle, A. and Jordan, N.: Visualizing the evolution of software using softchange, *Proceedings of thehe 16th internation conference on software dngineering and knowledge engineering (SEKE2004)*, pp.336–341 (2004).
4) Gutwin, C., Penner, R. and Schneider, K.: Group awareness in distributed software development, *Proceedings of the 2004 ACM conference on computer supported cooperative work (CSCW2004)*, New York, NY, USA, ACM, pp.72–81 (2004).
5) Herbsleb, J.D. and Grinter, R.E.: Architectures, coordination, and distance: Conway ’s law and beyond，*IEEE Software*, Vol.16, pp.63–70 (1999).
6) Jeong, G., Kim, S. and Zimmermann, T.: Improving bug triage with bug tossing graphs, *Proceedings of the the 7th joint meeting of the european software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, New York, NY, USA, ACM, pp.111–120 (2009).
7) Ohira, M., Koyama, K., Ihara, A., Matsumoto, S., Kamei, Y. and Matsumoto, K.: A Time-Lag Analysis Toward Improving the Efficiency of Communications among OSS Developers, *Proceedings of the 3rd International Workshop on Knowledge Collaboration in Software Development (KCSD2009)*, pp.49–62 (2009).
8) Schummer, T. and Haake, J.M.: Supporting distributed software development by modes of collaboration, *Proceedings of the seventh conference on european conference on computer supported cooperative work (ECSCW2001)*, Norwell, MA, USA, pp.79–98 (2001).
9) Storey, M.-A.D., Čubranić, D. and German, D.M.: On the use of visualization to support awareness of human activities in software development: a survey and a framework, *Proceedings of the 2005 ACM symposium on software visualization (SoftVis2005)*, ACM, pp.193–202 (2005).
10) W3C: Markup validation service, http://validator.w3.org/.