

Title	オープンソース・ソフトウェアの開発メカニズム： 基幹技術開示によるヒューマン・セキュリティ
Sub Title	
Author	林, 幹人(Hayashi, Mikito) 國領, 二郎(Kokuryō, Jirō)
Publisher	慶應義塾大学大学院政策・メディア研究科
Publication year	2004
Jtitle	総合政策学ワーキングペーパーシリーズ (Policy and governance working paper series). No.20
JaLC DOI	
Abstract	情報システムが社会の一つの重要な基盤を形成している現代において、情報システムを制御しその根幹をなすソフトウェアが、一部の企業によって提供され、その技術的な内容が秘匿されていることは、重大な社会的リスクとなりうる。ソフトウェアのオープンソース化は、こうした社会的リスクを解決しうるものの、Linuxのような顕著な成功事例がある一方で、その開発や普及を成功させることは難しい。本稿は、オープンソースの開発プロジェクトの事例を通じて、オープンソース化を成功させるための条件を検討するものである。本稿では、オープンソースの開発においてしばしば前提とされるボランティアの開発者とのコラボレーションに焦点をあて、開発者をコラボレーションへと動機づける側面と、開発参加を制約する側面から分析を試みる。そして、オープンソース開発におけるコラボレーションを成功裏に行うための条件を整理し、一般化への可能性を論じる。
Notes	21世紀COEプログラム「日本・アジアにおける総合政策学先導拠点」
Genre	Technical Report
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=BA76859882-00000020-0001

慶應義塾大学学術情報リポジトリ(KOARA)に掲載されているコンテンツの著作権は、それぞれの著作者、学会または出版社/発行者に帰属し、その権利は著作権法によって保護されています。引用にあたっては、著作権法を遵守してご利用ください。

The copyrights of content available on the Keio Associated Repository of Academic resources (KOARA) belong to the respective authors, academic societies, or publishers/issuers, and these rights are protected by the Japanese Copyright Act. When quoting the content, please follow the Japanese copyright act.

オープンソース・ソフトウェアの開発メカニズム

ー基幹技術開示によるヒューマン・セキュリティー

林 幹人*・國領二郎**

2004 年 2 月

21 世紀 COE プログラム

「日本・アジアにおける総合政策学先導拠点」

慶應義塾大学大学院 政策・メディア研究科

本稿は、2002 年 10 月から 2003 年 3 月にかけて、慶應義塾大学大学院経営管理研究科博士課程におけるマネジメント・システム特別実習の一環として行われた事例調査の結果に、その後の追加的な調査の結果を踏まえて大幅に加筆修正したものである。調査に際し、OSASK 計画の川合秀実氏には、多大なるご協力を頂いた。記して感謝申し上げたい。

* 慶應義塾大学大学院 経営管理研究科博士課程 (da026406@kbs.keio.ac.jp)

** 慶應義塾大学大学院 政策・メディア研究科／環境情報学部 (jkokuryo@sfc.keio.ac.jp)

オープンソース・ソフトウェアの開発メカニズム —基幹技術開示によるヒューマン・セキュリティ—

林 幹人・國領二郎

【概要】

情報システムが社会の一つの重要な基盤を形成している現代において、情報システムを制御しその根幹をなすソフトウェアが、一部の企業によって提供され、その技術的な内容が秘匿されていることは、重大な社会的リスクとなりうる。ソフトウェアのオープンソース化は、こうした社会的リスクを解決しうるものの、Linuxのような顕著な成功事例がある一方で、その開発や普及を成功させることは難しい。本稿は、オープンソースの開発プロジェクトの事例を通じて、オープンソース化を成功させるための条件を検討するものである。本稿では、オープンソースの開発においてしばしば前提とされるボランティアの開発者とのコラボレーションに焦点をあて、開発者をコラボレーションへと動機づける側面と、開発参加を制約する側面から分析を試みる。そして、オープンソース開発におけるコラボレーションを成功裏に行うための条件を整理し、一般化への可能性を論じる。

キーワード：オープンソース・ソフトウェア、ネットワーク・コミュニティ、プロジェクト・マネジメント、ソフトウェア開発、ヒューマン・セキュリティ

1 はじめに

情報システムが社会の一つの重要な基盤を形成している現代において、情報システムを制御しその根幹をなすソフトウェアが、一部の企業によって提供され、その技術的な内容が秘匿されていることは、重大な社会的リスクとなりうる。ソフトウェアの内容がブラックボックス化されていることは、情報システムが機能不全に陥った場合に、一部の人間にしか修正や回復ができない状況をもたらさうる。

こうしたリスクに対する一つの解法として、ソフトウェアのソースコードを開示するという方策がある。すなわち、ソフトウェアをオープンソース・ソフトウェア（以下、オープンソース）とすることで、誰もがその内容にアクセスし、問題を改修できる状態にしようとする方策である。社会的に重要性の高いソフトウェアがブラックボックスのまま他国の企業によって提供される状況を安全保障上の問題ととらえ、中国が、オープンソース化を推進する政策をとっていることは象徴的である。

しかしながら、オープンソース化については、Linux のような顕著な成功事例がある一方で、必ずしもうまくいかないケースが多い。Raymond (1998) は、Linux の開発プロジェクトに基づいてオープンソースの開発方式をまとめ、一見混沌としたバザールのようなオンライン・コミュニティから信頼性の高いソフトウェアが生み出される様子を見て、これを「バザール方式」と名づけた。しかし、この方式が普遍的なオープンソースの開発方式として広く適用できるかどうかは明らかではない。

本稿において、われわれは、現在オープンソースとして開発が続けられているオペレーティング・システム（以下、OS）である「OSASK」（おさすく）の開発プロジェクトの事例を取り上げ分析する。OSASK は、Linux と同じ OS であり、バザール方式に準じた開発方式を採用しているにもかかわらず、バザール方式が前提とするコミュニティとのコラボレーションは成功しているとはいいがたい状況にある。われわれは、OSASK プロジェクトのこれまでの取り組みを詳細に見ることで、オープンソースの開発におけるボランティア開発者とのコラボレーションを成功裏に実現するための条件を検討する。本稿の目的は、ヒューマン・セキュリティ上のリスクを回避しうるオープンソース化を成功させるための手がかりを提示することにある。なお、ここに含まれる事例は、2002 年 11 月から継続的に実施してきた OSASK プロジェクトのリーダーである川合秀実氏への面接及び電子メールによるインタビューと、公開資料の調査に基づいている。

2 事例

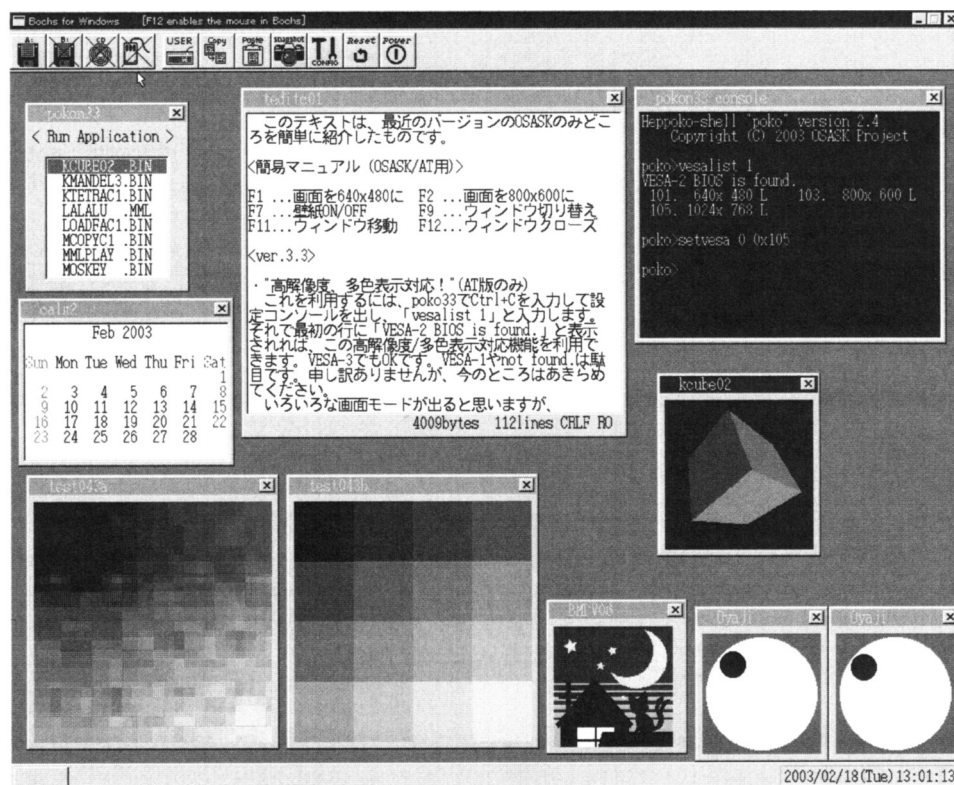
OSASK は、現在普及している OS に対する不満を克服するべく川合秀実氏によって企画されたユニークなコンセプトの OS であり、無償で配布されインターネット上にソースコードが公開されたオープンソースである。そして、この OSASK を開発し普及させようとするプロジェクトが「OSASK 計画」であり、川合氏を中心に、そのビジョンに共感した人々が自発的に参加協力するオンライン・コミュニティによって運営されてきた。

2.1 OSASK 計画の概況

(1) OSASK とは

OSASK は、「エミュレータ OS」というコンセプトを掲げる GUI（Graphical User Interface）タイプの OS である（図 1）。

図 1 OSASK の画面イメージ



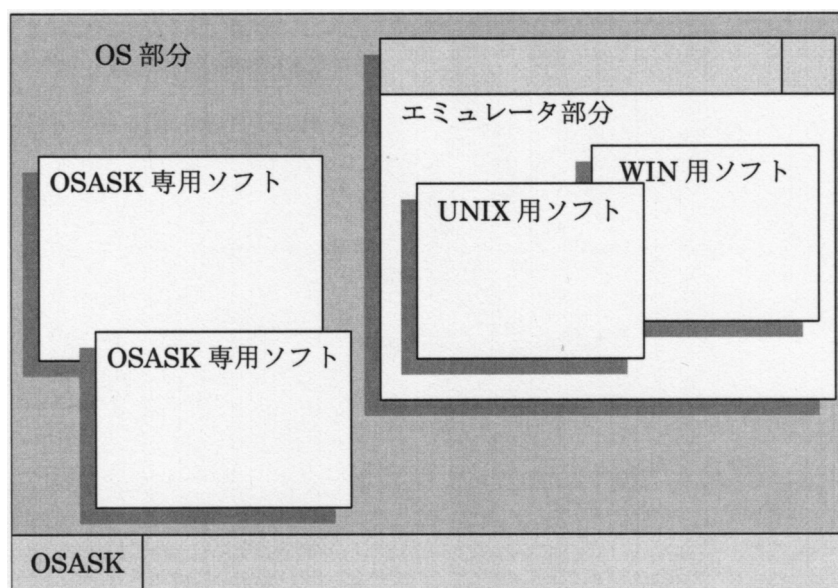
注) OSASK の画面で、専用アプリケーションが動作している状態である。 <http://osask.jp/picat3.3.html> より転載。

エミュレータとは、あるコンピュータの上で、別のコンピュータ用に開発されたソフトウェアを動作させるためのソフトウェアであり、例えば、Windows の画面の中に Macintosh の画面を再現し、その中で Macintosh 用のアプリケーションを動作させるような機能を持つものである。OS には Windows や Macintosh、UNIX など、さまざまな種類があるが、それらの上で動作するアプリケーションは OS の種類に依存する。つまり、通常、異なる OS の間には互換性がなく、ある OS の上では、他の OS 用のアプリケーションを使うことはできない。結果としてユーザーは、皆が使っているからといった理由で、欲しくもない OS を選択しなければならないといった事態が生じる。川合氏は、こうした状況に不満を覚え、多様な OS との間で互換性を確保し、数あるソフトウェア資産を共通に活用できるエミュレータのような OS を開発しようと考えたのであった。

そのため OSASK は、核となる OS 部分と、その上で動作するエミュレータ部分から構成される。

まず、OS 部分は、一般的な OS と同様に、ハードウェア資源の管理や基本的なユーザー・インタフェースなど、アプリケーションが動作するための基礎を提供する。もう一方のエミュレータ部分は、OSASK の OS 部分の上で動作し、他の OS をエミュレーションする部分である。OSASK とは、OS 部分とエミュレータ部分をあわせた全体のことである（図 2）。

図 2 OSASK の構造のイメージ



互換性の問題を克服するだけであれば、既存の OS を用いればよいが、その場合、処理速度が問題となる。川合氏は、既存の OS の多くが非効率な構造をしており、その結果、処理速度が遅く、不安定でさえあると考えていた。そのため、さまざまな OS をエミュレーションすることができたとしても、その上で動作するアプリケーションが快適に使用できるとは思えなかった。そこで、川合氏は、エミュレーションを前提としたきわめて効率的な構造の OS を開発することで、これを克服しようと思い立ったのである。

(2) プロジェクトの経緯

2000 年 3 月 31 日、川合氏は、OSASK のメーリング・リストに「開発再開宣言」を投稿し、2000 年の 4 月から本格的にプロジェクトが立ち上げられた（図 3）。OSASK の構想は 1998 年頃から練られていたが、未完成のソフトウェアがあったわけではなく、開発再開とは言うものの、まったくのゼロからのスタートと言ってよかった。

図3 OSASKの開発再開宣言（一部抜粋）

[OSASK 527] 開発再開宣言 • Subject: [OSASK 527] 開発再開宣言. • From: Hidemi KAWAI <kawai@imasy.or.jp> • Date: Fri, 31 Mar 2000 13:06:45 -0000 こんばんは、川合です。 4/1 から、OSASK の開発を再開します。 簡単な開発スケジュールは、以下のようにしようと思っています。 1. 毎月上旬に、OSASK の beta 版が OSASK-ML 購読者向けに公開されます。購読者のうち関心がある人は、beta 版を実行してみてください。不具合などを発見した場合は OSASK-ML 上で連絡してください。 2. 5/1 から、毎月 1 日に、OSASK のリリース版が一般向けに公開されます。これは、前月の月上旬に beta 公開したものを、報告を元に改良したものです。 -- 川合 秀実(KAWAI Hidemi) 川合堂社長 / OSASK 計画総指揮 / カーネル開発班 E-mail:kawai@imasy.or.jp Homepage http://www.imasy.or.jp/~kawai/

図4 OSASKの発展の経緯（開始～2001年6月）

公開日	バージョン	主な機能・改善内容
2000/05/01	OSASK/AT ver.0.0	簡単なグラフィックと、マウスのテスト。
2000/06/01	OSASK/AT ver.0.1	マルチタスクのデモ。
2000/07/01	OSASK/AT ver.0.2	音楽演奏(BEEP 音)とキーボードからの入力(USB キーボードは不可)。
2000/08/03	OSASK/AT ver.0.3	不便ながら、アプリケーションの実行ができる。
2000/09/01	OSASK/AT ver.0.4	実行可能なアプリケーションの幅が少し増えただけ。
2000/10/01	OSASK/AT ver.0.5	直接起動ディスクが生成できるようになった。
2000/11/01	OSASK/AT ver.0.6	内部整備(マルチタスク化への布石)。アプリケーション・ラウンチャーの見栄えが向上。
2000/12/01	OSASK/AT ver.0.7	アプリケーションがマルチタスクで起動できる。マウスは出なくなった。
2001/01/09	OSASK/AT ver.0.8	ANK フォント内包。設定は面倒だが、高解像度に対応。
2001/02/04	OSASK/AT ver.0.9	ビデオカードのプラグアンドプレイに対応。マウスを少しサポート。
2001/03/02	OSASK/AT ver.1.0	ウィンドウ内のグラフィック表示に対応。マウスサポートの強化。直接起動ディスクのフォーマットを標準的なものにし、インストール作業を簡単にした。
2001/04/01	OSASK/AT ver.1.1, OSASK/TOWNS ver.1.1	TOWNS 版初公開。AT 版はわずかなバグフィクスにとどまった。
2001/05/01	OSASK/AT ver.1.2, OSASK/TOWNS ver.1.2	ファイルシステムを少し構築。起動可能タスク数が大幅増加。おまけコンソールをサポート。
2001/06/01	OSASK/AT ver.1.3, OSASK/TOWNS ver.1.3	アプリケーションからフォントを定義できるようになった。キー入力を細かく検出できるようになった(メイク/ブレイクなど)。

注) <http://imasy.org/~kawai/osask/older.html> より作成。2004 年現在、バージョン 4.3 に達する。

プロジェクトの開始から、「完成度よりも、とにかく動くものを作る」というポリシーのもと、インストーラー付きの一般公開版を毎月リリースしてきた（図 4）。進捗がわずかしかなこともあったが、リリースを欠かすことはなかった。

ただし、OSASK は、当初はオープンソースではなく、クローズなソフトウェアとして、川合氏と大学時代の友人によって開発が進められた。OSASK がオープンソースとなったのは、2000 年末に、川合氏が、以前からフリーウェア作家として活動していたことを理由に、あるオープンソースのイベントで、パネル・ディスカッションのパネラーとして招待されたことがきっかけであった。「オープンソースのイベントに参加するのに、自分の作ったソフトウェアがオープンソースでないのはまずい」と考えた川合氏は、急きょ、オープンソースのライセンスを作成し公開した（図 5）。

図 5 OSASK のライセンス（一部抜粋）

川合堂ライセンス-01 ver.1.0	
2000.12.30 H.Kawai (川合秀実)	
0. 概要	
平たく言うと、「フリーソフトです。使用前使用後に対価を支払うことなく、自由に使えます。コピーしてもいいです。改変してもいいです。商業利用してもいいです。でもバグなどで損害が出てても責任はとれません。」ってことです。	
利用者が改変したり参考にしたりする人の利益のために、プログラム中で使っているアルゴリズムで著作者が将来特許をとることがあっても、特許料を要求したりはしないという保証もあります。	
1. 目的	
このライセンスで提供されるソフトウェアは、少しでも多くの人に利益をもたらす、ソフトウェア技術の進歩に少しでも貢献できればという目的で公開する。	
2. 趣旨	
このライセンスは、著作権を放棄するものではない（独占的にコピーする権利は放棄している）。利用者はこのソフトウェアの一部または全部を自由にコピーし、再配布することができる。利用に際して対価を要求しない。解析、改変も対価なしに認める。	
このライセンスが適用されるソフトウェアの利用について、商業的な利用も無条件で認める。そのまま有償で販売しても構わない。	
このライセンスが適用されるソフトウェアの一部または全部を元にして作成されたソフトウェア（以降、派生物と称する）に対し、どんなライセンスを付与してもよい。すなわち、派生物がコピー禁止であってもよいし、派生物が有償でしか配布されなくても構わない。もちろん無償であってもよい。派生物に対する著作権は、派生物を生成した者に帰し、このライセンスが適用されるソフトウェアの著作者が派生物に対して著作権を主張することはない。	
派生物のドキュメント中に、元にしたソフトウェアの著作者を紹介する義務はない。この文は、もちろん、紹介することを禁止するものでもない。	
派生物の公開に際して、元にしたソフトウェアの著作者に確認を取る義務はない。この文は、もちろん、確認を禁止するものではない。	
著作者は、ソフトウェアの質を保証しない。したがって、このソフトウェアで被害を被ったり、期待した結果が得られなくても、著作者は責任を負わない。	
このライセンスが適用されるソフトウェアで使われている技術については、事前に著作者に許された者以外が特許を取得することは禁止する。新たな技術を加えた派生物を生成し、その追加された部分の特許をとることは認める。著作者がソフトウェア中の技術に対して後から特許をとることはありうるが、派生物やこのソフトウェアの利用に対して特許料やその他の対価を求めることはないことを保証する。この保証は、特許取得前に生成された派生物だけでなく、特許取得後に生成された派生物にも適用される。	
解析結果をまとめて特許をとることは事前の著作者の許可が必要だが、特許をとること以外については何ら制限はない。	

注）全文は、<http://www.imasy.org/~mone/kawaido/license01-1.0.html> を参照。

このライセンスは、ソフトウェアの利用、改変、再配布、商業利用などをすべて自由にすることを

宣言するものであり、このライセンス下のソフトウェアから派生するソフトウェアに一切の制約を設けないという、著作権の放棄と同等の意味をもつものである。一般にオープンソースのライセンスとして適用される GNU GPL (General Public License: 一般公衆利用許諾契約) や FreeBSD コピーライトは、自由なソフトウェア・ライセンスと言いながらもいくつかの制約を課している。そのため川合氏は「完全に自由にしたい」という理由から独自のライセンスの適用に踏み切った。OSASK 開発のコミュニティでも、自由で気楽なライセンスということで評判は良かったものの、一方では、他者による悪用について心配する声もあがっていた。しかし、川合氏は、「僕の作ったソフトが世の中のためになるんだったら、自由にやってください」、という立場をとった。

開発再開宣言の翌月の 2000 年 5 月に、川合氏はバージョン 0.0 を公開したが、それは、画面表示を司る VGA (Video Graphics Array) のコントロールを試すために開発されたものであり、表示された画面の中でマウスカーソルを動かせるというだけの単なる実行ファイルに過ぎなかった。また 6 月には、コンピュータが複数の仕事を並行して実行できるようにするマルチタスクを試すために、起動と同時に三つの同じ処理が立ち上がるだけというプログラムを作成した。このように、初期のバージョンの OSASK は、OS というよりも、ハードウェアをコントロールするための技術的なテスト・プログラムでしかなかった。しかも、それらには終了方法すらなく、終了するには電源を落とすしかないという代物であった。OSASK が、オープンソースとして公開される 2000 年末頃には、マルチタスクで複数のアプリケーションが起動できるようになっていたが、OS としてはまったく未熟であった。そのすぐ後の 2001 年 3 月には、バージョン 1.0 が公開されたものの、ファイル・システムすら実装されていなかった。

しかしながら、OSASK は、少しずつではあるが着実に発展を遂げてきた。バージョン 4.3 に達した現在、いまだ OS としては十分とはいえないものの、既にファイル・システムが実装され、800 × 600 の解像度にも対応し、マルチタスク・マルチウィンドウが実現されていた。

これまでに、OSASK を構成する OS 部分とエミュレータ部分のうち、OS 部分の基本的な機能のみが開発されていたが、エミュレータ部分の開発にはいまだ着手されておらず、「エミュレータ OS」の構想は具体化されるに至っていなかった。ただ、エミュレータ部分がなくても OSASK 専用のアプリケーションは実行可能であり、OSASK コミュニティのメンバーによって、テキスト・エディタや、電卓、ストップウォッチ、ゲームなどのアプリケーションが開発されていた。

2.2 OSASK の内部構造

(1) モジュール化

これまで OSASK の開発は、ある程度開発しては大幅に作り直し、またある程度開発してはほとんどを書き換えるというプロセスを経てきた。例えば、開発が再開されて 4 ヶ月を過ぎた 2000 年の 8 月、ソースコードの全体に及ぶ最初の修正が行われた。それまで、OSASK のコア部分は一続きのプログラムでできていたため、これを小さなモジュール単位に分割したのである。

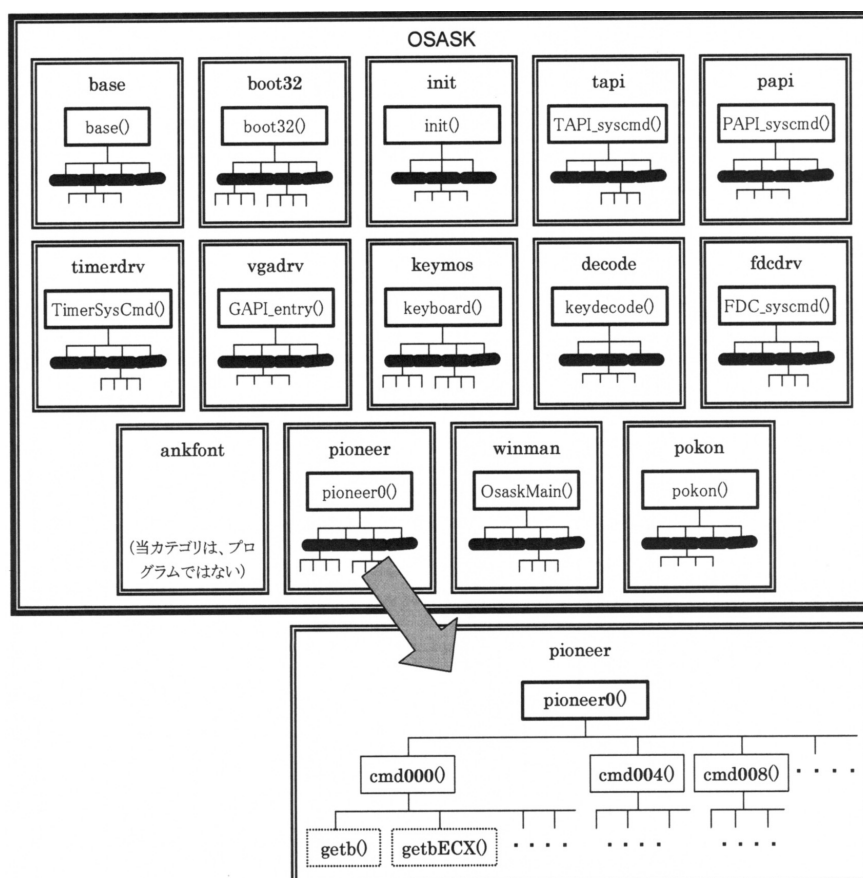
ソフトウェアが一続きのプログラムでできていると、その一部の修正がソフトウェア全体に影響する

ために、規模が大きくなるにつれてメンテナンスの負荷が増大する。モジュール化を行えば、個々のモジュールのインタフェース部分を変更しない限り修正の影響をモジュール内部で完結させることができるため、そうした負荷を下げるができる。また、モジュール化は同時に、分散環境におかれた開発メンバーが、他の開発メンバーに影響を与えることなく独立に並行して共同開発することを容易にする。OSASK の最初の大規模な改修は、こうした効果が期待されるモジュール化を行うためのものであった。

現在までに、OSASK の内部は、14 のカテゴリに分類され、各カテゴリは、階層構造をした関数のグループによって構成されていた (図 6)。例えば、API (application programming interface) を司る pioneer グループは、pioneer0() という関数を最上位にして、下位に約 50 の関数を持ち、さらにその下位にも約 50 の関数を持っていた。OSASK トータルでは、そうした関数の数は数百に及んでいた。

各関数は、その呼び出し方法などインタフェースに関する仕様が規定されており、一部についてはドキュメントにまとめられていた。ドキュメント化されていないものでも、その関数を使用しているプログラムを参照することでその使い方を知ることができた。OSASK のモジュールは、このような構造をとることによって、そのインタフェース部分を変更しない限り、それ以外のモジュールとは独立に修正することが可能であった。

図 6 OSASK の内部構造



関数の階層構造は、あくまでもグルーピングのためのものであって、関数同士の使用関係と一致するものではなかった。すなわち、下位の関数であっても、上位の関数を呼び出すことができ、これによって再利用性を高めようとしたのである。したがって、使用関係については、階層構造というよりもネットワーク構造に近かった。

一方で、ネットワーク構造をとることによって、構造が複雑になることが懸念される。そこで、構造が必要以上に複雑になることを避けるために、他のグループ、例えば、winman グループの関数が、pioneer グループの関数を呼び出して使用する場合には、最上位の pioneer0() にしかアクセスできないルールが設定されていた。こうすることで、関数同士が網の目状に絡み合い、複雑化することを回避していたのである。

川合氏は、モジュール化によって、ソースコードの中から問題箇所が特定しやすくなるという効果と、一部の修正がソフトウェア全体に波及しないという効果が得られていると実感していた。ただ、モジュール化によって、共同開発が容易になるという効果については疑問があった。もとより開発メンバーが少ないこともあったが、開発に参加しているメンバーであっても、あるモジュール全体を開発したり改善したりするということではなく、その一部分だけを改良するにとどまっていたからである。

また、コンセプトがユニークであるために最終的な完成イメージを描けない OSASK にとっては、モジュール化すること自体が容易ではなかった。つまり、開発の初期段階の完成イメージにしたがって分割したモジュールでは、後から生じてくるニーズのすべてに対応できるとは限らないのである。そして、完成イメージが修正され、モジュールの分割方法あるいはモジュールの関係構造の見直しが要求される場合には、既存の膨大な数のモジュールをチェックし、多大な修正作業を行う必要が生じた。

その結果、OSASK の開発においては、ある程度開発しては大幅にリフレッシュされる、といったことが繰り返し行われてきた。しかしながら、そのような大幅なリフレッシュの頻度も少なくなり、次第に安定的な開発が行われるようになりつつあった。

(2) プログラミング言語

代表的なオープンソースの開発においては、高級言語と呼ばれる C 言語や C++、Java などが用いられていることが多い。しかし、OSASK は、その処理速度やサイズなどの効率性を最も重視した OS であるため、オープンソースとしてはめずらしく、主要な開発言語としてアセンブリ言語が採用されていた。アセンブリ言語は、機械語を、人間が理解しやすい記号にほぼ一対一の関係で置き換えたものである。

人間によって書かれたプログラムは、コンピュータが直接理解できる機械語に翻訳されなくてはならないが、高級言語で書かれたプログラムの場合、その一つの処理命令は通常、複数の機械語命令に対応している。そのため、高級言語を使えば、書かなくてはならないプログラムの行数を減らすことができる。ある報告によれば、同じ処理のプログラムを書く場合、C 言語はアセンブリ言語の半分以上の行数で済むとされる。すなわち、アセンブリ言語を使うことは、開発効率を低下させることを意

味する。

また、アセンブリ言語は、ハードウェアへの依存度が高く、その特性に応じたプログラミングをしなければならないために、使用するハードウェアに関する深い知識が必要となる。高級言語であれば、通常、そのソースコードを機械語に変換するコンパイラやインタプリタといったシステムが、ハードウェアの違いを吸収するため、ハードウェアに関する知識はそれほど要求されない。すなわち、高級言語を使うことで、ハードウェアに対する汎用性の高いプログラムを作成することが容易になる。

しかしながら、高級言語から機械語への翻訳は、コンパイラやインタプリタによって機械的に処理されるため、多くの場合、生成される機械語は冗長で無駄の多いものとなる。これに対し、アセンブリ言語の場合は、高級言語のような便利な書き方はできないものの、機械語の命令に対応した機能範囲の小さな命令を緻密に組み合わせることで、無駄のない、効率的なプログラムを書くことができるのである。

そこで川合氏は、アセンブリ言語の長所を活かしつつ開発効率を高めようと「ASKA」というC言語に似たアセンブリ言語を開発し OSASK の開発に使用した。C言語に似ているとは言っても、やはり基本的にはアセンブリ言語であり、高級言語が可能にする開発効率には到底及ばなかったものの、その書き方や言語体系は高級言語に近く理解しやすいものとなっていた（図7）。その結果、一般的なアセンブリ言語よりもプログラム作成やデバッグの能率が上がり、習得も容易になることが期待されたが、実際、川合氏は、それが実現されていると感じていた。

図7 ASKA と一般的なアセンブリ言語の比較

一般的なアセンブリ言語	ASKA
<pre>XOR EAX, EAX XOR ECX, ECX LABEL: ADD EAX, ECX INC ECX CMP ECX, 10 JLE LABEL</pre>	<pre>int sum == EAX, i == ECX; sum = 0; i = 0; do { sum += i; i++; } while (i <= 10);</pre>

注) 上記の両方のソースコードは、完全に同じ機械語を出力する。

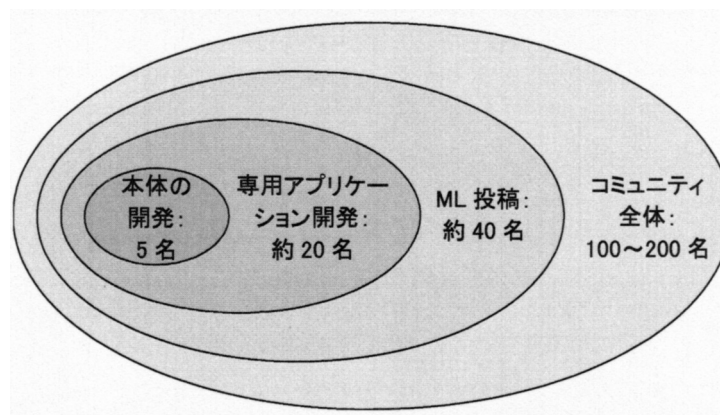
こうして、OSASK はその高い効率性を実現するために、基本的には ASKA によって開発されてきたが、C言語を使用することもできた。実際、ASKA で書かれているのは、OSASK 全体の約7割であり、ロジックが複雑で難易度の高い箇所や、頻繁に更新される部分については、C言語が使用されていた。川合氏は、OSASK の高い効率性を維持しながら開発を進めていきたいと考えており、特に理由がない限り ASKA を使用することにしていた。既存の OS とは機能も異なるため単純な比較はできないものの、OSASK は、バージョン 4.3 で約 75KB というきわめて小さなサイズを実現していた。

2.3 OSASK コミュニティ

OSASK コミュニティのメンバーは、川合氏の大学時代の友人を核として、展示会などのイベントに参加したり、雑誌に掲載されたりするたびに少しずつ増えてきたが、その数は決して多くなかった。川合氏は、「一人で開発するという最悪のケースを考えれば、これだけ自発的な参加者が集まってきてくれることはありがたい」と言いながらも、やはりメンバーの数はまだ十分ではないと考えていた。実際、OSASK のホームページには、「求人」として、プロジェクトにおいて手伝って欲しい案件が多数掲載されていた。

コミュニティのメンバーはきわめて流動的であり、その規模を正確に特定することは難しいが、川合氏は、何らかの形でコミュニティに関与している人々の数は100名から200名にのぼるとみていた。また、一般に、メーリング・リストには、発言を投稿しないで読むだけのメンバーも多数含まれるが、プロジェクトの開始から2003年5月までの約3年間に、OSASK のメーリング・リストに投稿したメンバーの数は約40名を上回っていた。さらに、OSASK の「おすすめパック」として公開されているOSASK 専用のアプリケーションのクレジットを見ると、約20名の協力者が存在した。そして、OSASK 本体の開発には、これまでに5名が協力してきたという。したがって、OSASK コミュニティの規模は、最低でも50名近く存在し、その内20名程度が積極的な貢献を果たしていると思われた(図8)。

図8 OSASK コミュニティの規模と構成



注) 厳密な把握は難しいが、川合氏の主観的評価と複数の資料に基づいて評価した。

川合氏と開発メンバーとのやり取りは、メーリング・リスト、掲示板、個人的なメールで行なわれてきた。基本的には、ユーザーが、OSASK を使用し、その問題点や要望を川合氏に送り、川合氏が納得すれば、その実現に向けて検討するというやり方がとられた。

コミュニティを運営するにあたって、川合氏は、OSASK 計画に参加してくれるメンバーの自発性を最も尊重してきた。そのため、開発メンバーからのプログラムの送付や、ユーザーやその他からの問い合わせには、必ず誠意を持って回答するようにした。メンバーから川合氏に送られたプログラム

に関しては、「せっかくの貢献を無碍に拒否することはできない」と考え、できるだけ取り込もうという立場で臨んだ。時には、送付されてきたプログラムをレビューし、問題のある箇所などを修正して、OSASK に組み込んだこともあった。プログラムを送付したメンバーからプログラムが修正されていることを指摘されることもあったが、川合氏は、そのつど修正理由について説明した。

川合氏とメンバーとの間で意見が衝突したときには、お互いが納得するまで議論が行われた。そして、川合氏自身になるほどと思った点については、折れることにしていた。ただ、川合氏のこだわりから、どうしても譲れない部分もあり、その点については川合氏の考え方で開発が進められた。それは、川合氏が「基本的に OSASK は、僕自身の夢」と語るように、最悪の場合には、自分一人で開発にあたってもいいという考えがあったからである。

そして、もし自らの方針と合わないときには、川合氏は分派することをむしろ奨励した。分派とは、フォークとも呼ばれ、当初のプロジェクトを主流とすれば、それを改変して傍流というべきバージョンを作ることである。分派によって開発者が分散すると、コミュニティの勢力が衰えるため、通常、オープンソース・コミュニティでは避けるべき行為と見なされる。川合氏が、分派を奨励する理由は、川合氏自身の夢である OSASK についてのこだわりを誰にも譲りたくないことと、同時に他のメンバーにもそれを強制したくないという思いがあったからである。つまり、その分派こそが将来の主流になるのかも知れないのであって、それをつぶすようなことは避けたいと考えたのである。そうなるくらいなら、「むしろ分派して、お互いに競争したほうがいい」というのが川合氏の考えであった。

また、川合氏は、潜在的な開発メンバーであり利用者である一般の人々からの問い合わせにも、できる限り誠意を持って対応してきた。問い合わせには、例えば「OSASK って何ですか?」といった質問をはじめ、「低級言語を使うなんて時代に逆行している」「今時、効率性よりも使いやすさの方が重要なのではないか?」といった OSASK のコンセプトに対する意見や、「途中で頓挫するのでは?」「構想が大きすぎるんじゃないか?」と、その実現を疑問視するものまで存在した。さらには、「考え方には共感できるが、いつ完成するかわからないものに、時間をかけて協力したくない」というものまであった。

OSASK に関する問い合わせは多数にのぼり、内容也多岐にわたっていたが、川合氏は、それが、OSASK のコンセプトが極めてユニークであり、理解することが難しい結果であると考えていた。そのことを示すように、OSASK 計画のホームページには、「OSASK Q&A」「OSASK への意見と回答」「よくある誤解」「OSASK の開発方針」といった OSASK のコンセプトの理解を助けるためのメニューがいくつも並んでいた。さらに、個々の問い合わせにも、川合氏は、一件ずつ丹念に対応し、理解を促そうと心がけた。

川合氏は、そのように丁寧に対応することで、同時に、OSASK コミュニティの理解をも促し、いつかはコミュニティのメンバーが、問合せへの対応を自発的に行ってくれることを期待した。実際、新規の問い合わせには、数回は川合氏自身が答えなくてはならなかったものの、過去に寄せられた問合せに対しては、次第にコミュニティのメンバーが自発的に回答するようになりつつあった。川合氏は、そのおかげで、問い合わせ対応の負荷は徐々にではあるが確実に減少していると感じていた。

2.4 OSASK 計画の問題

OSASK 計画の最大の問題は、その開発進度が遅く、その完成時期がまったく見えないことであった。これまで川合氏は、今後の開発の見通しをまとめ、1～2ヶ月先までの具体的な計画から1年以上先の予定までを、OSASK 開発のロードマップとして公開してきた。しかし、川合氏は、その計画が予定通り実行されるかどうかについて自信がないばかりか、OSASK の完成時期に至ってはまったくといっていいほどわからなかった。実際、このことを示すように、そのロードマップは「開発計画」ではなく、OSASK の「開発予想」と名づけられていた。

川合氏は、開発が計画的に進まない理由の一つは、OSASK の全体像あるいは完成図が明確でないからと考えていた。オープンソースの開発においては、しばしば、ユーザーや共同開発者とのやり取りの中で、その完成イメージやその方向性が決められることがあり、そもそも明確な開発計画を立てること自体が容易ではない。OSASK もその例外ではなかったが、OSASK は特に、そのユニークで新しいコンセプトのためにそういったイメージを描くことが難しく、手探りで開発を進めるしかなかった。その結果、発展の過程で構造上の問題や限界にぶつかることを避けられず、これまで幾度となく大幅なりフレッシュが必要になったと考えられた。

また、開発の速度が遅い理由は、コミュニティの規模が拡大しかなりの人々が関心を寄せるようになる一方で、OSASK の本体の開発には少数の開発者しか協力していないという状況があった。OSASK コミュニティには、少なくとも20名～50名のメンバーが積極的に協力し、OSASK の本体の開発にも5名が貢献してきたが、その内実はOSASK 本体の8割以上を川合氏が開発しており、事実上、川合氏がほぼ単独で開発してきたと言ってよかった。つまり、OSASK コミュニティの活性化によって、川合氏のコミュニティ運営にかかわる作業が増える一方で、OSASK 本体の開発は、相変わらず川合氏一人にゆだねられていたのである。

その原因として、OSASK の本体の開発に参加することが非常に難しいという状況があった。つまり、OSASK の開発を行うためには、川合氏が開発した ASKA を使用しなければならないばかりか、OSASK の効率性と高速処理を維持するために、アセンブラやハードウェアに関する深い知識に基づいて ASKA を高度に使いこなす必要があったのである。

事実、遅々とした開発状況に一抹の懸念を抱いた OSASK のコミュニティ・メンバーからは、こうした状況を打開するために「とりあえずC言語で開発してはどうか」、そして、「後からアセンブリ言語で書き換えてもよいのではないか」という声があがっていた。つまり、OSASK の効率性の追求は後回しにして、その基本的な機能を完成に近づけてはどうか、というのである。それは、時間と労力をかけずに実稼動するプロトタイプを開発し、これによって全体像や完成イメージを共有しやすくしようという提案であった。普及率の高いC言語を利用することで新たな協力者が参加しやすくなり、OSASK コミュニティの勢力が増強されることで開発スピードが向上するのでは、と考えたのである。

しかし、川合氏は、あくまでもアセンブリ言語での開発にこだわりたいと考えていた。なぜなら、効率性の実現は OSASK を最も特徴づけるポイントであり、それ以外の機能や GUI などは、既存の OS よりも間違いなく劣っていたからであった。そして何よりオープンソースとして公開している「効

率性の高い OSASK」それ自体こそが、広告であり看板であり、川合氏の「夢」を体現したものにはかならなかったからである。

また、アセンブリ言語を使用しているからこそ、OSASK 計画のビジョンに強く共感でき、かつ、高度な技術知識を持ったメンバーが集まってきていると川合氏は感じていた。実際、コミュニティの中には「C 言語や Java のような高級言語を使うなんて技術者として甘い」とまで言うメンバーさえいた。

川合氏は、C 言語での開発に一時的に切り替えることは、OSASK の開発速度を上げるために現時点でとりうる最良の方法であると認識していた。しかしながら、それは、OSASK の基本コンセプトの一つである「効率性」の維持を諦めることを意味し、川合氏にとってはやはり難しい選択であった。

2.5 OSASK 計画の将来

2004 年 1 月、OSASK はバージョン 4.3 に達し、OS として一通りの機能を備えつつあった。しかし、「エミュレータ OS」という構想には、いまだ着手すらされていない状況であり、完成までには当分の時間がかかることが予想された。

また、プロジェクトとしては、2002 年には特別認可法人情報処理振興事業協会の未踏ソフトウェア創造事業未踏ユースに採択されたことをはじめ、2003 年にはプロジェクト開始以来はじめてオフ会が実施されたことや、大規模な展示イベントへの出展、複数の雑誌への掲載、川合氏による慶應義塾大学理工学部での講演、といった出来事があった。

これらの出来事は、OSASK の知名度を高め、コミュニティを活性化させることが期待される一方で、川合氏の負荷を高め、OSASK の開発をさらに遅らせてしまうことも懸念された。川合氏がほとんど一人で開発を行っている状況では、川合氏が開発作業に注力できないことは、開発を停滞させることを意味した。川合氏は、これまでどおりにプロジェクトを進めていくか、それとも何らかの別の方策をとるか、OSASK 計画の将来を左右する岐路に立たされていた。

3 分析

われわれは、OSASK の開発プロジェクトの事例を失敗の事例としてとりあげるつもりはない。プロジェクトは継続しており、OSASK も日々発展している。われわれの問題意識は、なぜ OSASK の開発プロジェクトにおいては、他の代表的なオープンソースの開発プロジェクトに比べて、ボランティアの開発者によるコラボレーションが生じにくいのか、という点にある。

しかしながら、他の代表的なオープンソースの開発プロジェクトとは、プロジェクトの開始されたタイミング、継続期間、環境などさまざまな要因が異なるため、単純に比較することは難しい。そこで、本稿では、なぜ OSASK の専用アプリケーションの開発には 20 名以上の開発者が協力し実際に成果をあげているのに、OSASK 本体の開発においてはその発展に寄与するようなコラボレーションがほとんど生じていないのか、という問題を考察したい。ここでは、オープンソース開発におけるコ

ラボレーションに人々を引き付ける動機づけ要因と、人々のコラボレーションを制約する要因を検討する。

3.1 コラボレーションの動機づけ要因

Raymond (1998) によって提唱されたバザール方式、あるいは、それに準じたオープンソースの開発方式は、ボランティア開発者によるコラボレーションを前提とする。そして、こうしたコラボレーションが生じる理由は、開発者の「関心」というキーワードによって説明される。つまり、ボランティアの開発者は、基本的には自己の関心に基づいて開発に協力するとされ、仲間からの名声を獲得するために貢献し、獲得された名声を失わないように振舞うとされるのである (Raymond 1998)。

Feller and Fitzgerald (2002) は、さらに詳しく、オープンソースの開発プロジェクトの背後に存在し、人々にオープンソース開発を行わせるモチベーションを、すでに多数行われている先行研究をもとに整理した。彼らによれば、それは、社会政治的モチベーション、技術的モチベーション、経済的モチベーションの3つに分類できる。

社会政治的モチベーションとは、他者のために役立ちたいあるいは仲間から高い評価を得たいとする欲求や、プログラミング自体やコミュニティとのかかわりを楽しみとし、これを行いたいといった欲求を満たそうとする意向である。あるいは、ソフトウェアは自由であるべきというイデオロギーや、大規模なソフトウェア企業への反発といったものから生じるモチベーションも、これに含まれる。

技術的モチベーションとは、オープンソースという手段を用いて、ソフトウェアの開発を成功させようとする意向である。オープンソースという手段を用いることで、他の開発者からのレビューを利用し、優れた開発者が持つ最先端の技術を活用し、一方で退屈なテストや文書作成などの作業を多くのユーザーと共有することができる。こうした利益が得られるとすれば、ソフトウェア開発を成功させるために、オープンソースという手段を選択しようとする動機づけが働く。

経済的モチベーションとは、オープンソースを開発する、あるいは、プロジェクトに参加することによって、開発技術を習得しようとしたり、名声を得ることで将来のキャリアに役立てようとしたりする意向である。また、開発したソフトウェアを利用することによって得られる便益への期待も、開発者を動機づけるかもしれない。さらには、オープンソースを用いてなんらかのビジネスを企て、より直接的に経済的な利益を得ようとすることもあるだろう。

オープンソースの開発方式が、ボランティアの開発者とのコラボレーションを前提とするならば、オープンソースの開発プロジェクトを成功させるための一つのカギは、そこに参加することが、人々のそうした欲求を満たしうることを示すことであると考えられる。ここでは、Feller and Fitzgerald (2002) の分類に基づいて、OSASK の開発プロジェクトにおけるモチベーションのあり方を、OSASK 本体の開発と専用アプリケーションの開発に分け、相対的に評価する。

まず、社会政治的モチベーションという点では、OSASK は、無償で提供されるオープンソースであるから、開発に参加し発展に寄与することは、多くの人々のために役立ちたいという欲求を満たしうる。さらに、OSASK は、既存のいくつかの OS が相互に互換性がないことによって不便な状況を

克服するために構想された OS であり、現在のコンピュータ利用者の問題を解決するという大儀がある。それはまた、クローズなソフトウェアを提供する企業を問題視し、ソフトウェアは自由であるべきとするフリー・ソフトウェアないしオープンソースのコミュニティがしばしば唱導するイデオロギーを支持するものである。

これらの点で、OSASK 本体の開発と専用アプリケーションの開発には、ほとんど違いはないと考えられる。両者は、相互依存的な関係にあり、どちらか一方だけが成功しても、ソフトウェア製品としては十全ではなく、社会政治的な欲求は充足されない。したがって、OSASK 本体の開発と専用アプリケーションの開発における社会政治的モチベーションは、ほぼ等しいと見ることができる。

また、OSASK は、当然、オープンソースという手段をもちいてソフトウェア開発を成功させようとしているのであるから、技術的モチベーションがコミュニティのメンバーを動機づけていることは明らかである。実際、OSASK プロジェクトにおいても、他の代表的なオープンソースと同じく、テストや文書作成、問合せ対応などを多くのコミュニティ・メンバーと共同して行っているし、他の開発者によるレビューの機会も提供されている。

したがって、技術的モチベーションという点においても、OSASK 本体の開発と専用アプリケーションの開発の間に大きな違いはないと考えられる。OSASK の専用アプリケーションとして配布されているものも、その多くが OSASK と同じライセンスが適用されているか、少なくともソースコードが公開された状態にある。また、OSASK 本体と専用アプリケーションの開発が分担されているという関係自体が、オープンソースという手段によって可能になるところも多く、技術的なモチベーションの産物と評価できよう。

最後に、経済的モチベーションという意味でも、OSASK はコミュニティのメンバーを動機づけていると考えられる。OSASK は、エミュレータ OS というコンセプトを掲げており、完成すれば既存の互換性のない OS 用に開発されたソフトウェア資源を活用できることが期待される。この期待は、人々に、OSASK を発展させ、完成に近づけるよう貢献しようとする意向に働きかけうる。また、OSASK によって、将来何らかのビジネスを始め、いずれ直接的な経済的利益を得ることができるかもしれないという期待もある。さらに、OSASK の発展に寄与し、仲間からの名声が得られれば、そのことが将来のキャリアに有利な結果をもたらすことも考えられる。もちろん、OSASK のコミュニティは代表的なオープンソースのそれと比べて小規模であるために、開発者を動機づける力は強くない。しかし、代表的なオープンソースも、そのプロジェクトの立ち上げの際には、現在の OSASK と同じように知名度は低かったことを踏まえれば、この点は時間が解決するかもしれない。

しかしながら、プログラミングなどの技術的スキルを身に付けようとする人々にとっては、OSASK 本体の開発と専用アプリケーションの開発の間に、経済的モチベーションについての無視できない差があると考えられる。OSASK 本体には、その効率性と高速処理を実現するために、独自のアセンブリ言語 ASKA が使用されている。これは、プログラミングやデバッグの効率をあげるために川合氏によって開発された言語であり、一般的なアセンブリ言語よりも読みやすくなっている。しかし、開発技術を習得したいという経済的モチベーションによって開発に協力しようとする開発者に

としては、汎用性のない特殊な技術を身に付けることは合理的ではない。たとえそうした技術を習得しても、OSASK のプロジェクトが終わってしまえば、他で使用することはできないからである。また、将来のキャリアという点でも、ASKA を使いこなせることが、有利に働くとは考えにくい。コミュニティでの名声を獲得しても、一般的な技術を使いこなせる場合と、そうでない技術を使いこなせる場合とでは、評価も異なりうるのである。

したがって、経済的モチベーションの観点では、OSASK 本体の開発は、より一般的な技術を使用できる専用アプリケーションの開発よりも、ボランティアの開発者を動機づける力が弱いと判断できる。OSASK の開発プロジェクトにおいて、専用アプリケーションの開発には 20 名以上のボランティアが協力し成果をあげているにもかかわらず、OSASK 本体においては成果として結実した協力がほとんど得られていない状況は、経済的なモチベーションの違いに一つの原因があると考えられる。

3.2 コラボレーションの制約要因

オープンソースの開発プロジェクトを成功させるためには、ボランティアの開発者の関心を喚起し動機づけることによって、コラボレーションへと向かわせることが肝要である。しかしながら、それと同時に重要なことは、コラボレーションを制約する要因をコントロールし、開発に参加しやすい環境を整えることであると考えられる。

ここでは、コラボレーションを制約する要因として、オープンソース開発における情報処理活動の側面に焦点をあてる。情報処理活動とは、コミュニティ・メンバーによる意思決定や情報収集、伝達、調整といった活動を指す。Simon (1996) によれば、人工物は、それをつくる人間の認知能力、つまり、情報処理能力によって制約を受ける。そのため、人工物は通常、認知能力の限界を克服するためのメカニズムを有する。人々によって形成される組織も人工物であり、ルールや階層構造、目標などのメカニズムを使用することで人間の認知能力を節約し、組織が制御不能に陥ることを防いでいる。

ソフトウェア開発の組織においても、情報処理能力の限界からくる問題が指摘されている。ブルックスの法則として知られる、遅れているソフトウェア開発プロジェクトへの要員追加がさらにプロジェクトを遅らせる (Brooks 1995)、という命題は、要員を追加することが、作業の再配分の作業とそのための中断、新しい要員の訓練、新たに必要となる相互コミュニケーション、といった情報処理負荷の増大を引き起こすことを指摘したものといえる。

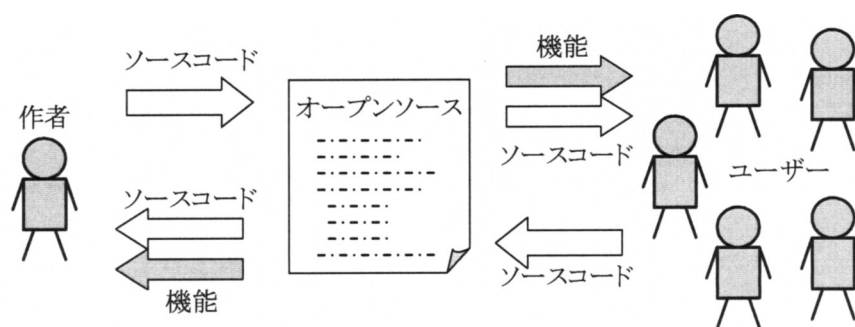
また、ソフトウェア工学の手法も、情報処理能力の限界を克服するべく発展を遂げてきた側面がある。例えば、ソフトウェアの構造化設計からオブジェクト指向設計への流れは、ソフトウェアを小さなサブシステムに分解し、サブシステム間をシンプルなインタフェースによってつなぐことで、ソフトウェア全体としての複雑さを削減し、それぞれのサブシステムを複数の開発者が並行して独立に開発することを可能にするものである。

このように、ソフトウェア開発という対象を考察する場合、情報処理活動という側面から接近することは有効であると思われる。さらに、オープンソースは、オンサイトでのソフトウェア開発よりもコミュニケーション手段に制約の多いインターネット上という環境において開発が行われるため、メ

ンバー間の調整や合意形成といった情報処理活動の負荷が、コラボレーションの成否により強く影響を及ぼすことが予想される。したがって、ここでは、オープンソース開発プロセスにおける情報処理活動という観点から、OSASK 本体の開発と専用アプリケーションの開発との違いを検討する。

オープンソースの開発プロセスは、まず、作者や中心的な役割を担うメンバーによって、ソフトウェアのソースコードが公開されることから始まる。つぎに、それを利用した人々は、その作者の開発意図やソフトウェアの機能を理解する。あるいは、ソースコードを参照することで、それがどのような意図で、どのように開発されているかを知る。そして、ソフトウェアに何らかの問題や不足があることを認識すると、それを改善すべくソースコードという形で具体化し、修正プログラムとして作者に送付する。作者は、その機能やソースコードを確認してその開発者の意図を理解し、採用が認められれば公式バージョンとしてまた公開する。オープンソースはこうして、作者とコミュニティのやり取りを通して開発されていく（図9）。

図9 オープンソースの開発プロセス



注) 作者とは、リーダーやコアメンバー等の中心的なメンバーを意味し、ユーザーとはオープンソースの利用者や協力者、ボランティアの開発者を含む。「機能」の矢印は、オープンソースの使用を通じて、その有用性や作者の開発意図が理解されることを意味する。

オープンソースの開発プロセスをこのようにとらえると、オープンソース自体が、コミュニティ・メンバーのコラボレーションを媒介していることが理解できる。ソフトウェア企業による開発プロジェクトにおいても、ソフトウェアが修正されたにもかかわらず仕様書は更新されないという状況がよくあることが知られているが、その結果、ソフトウェアのソースコードから直接仕様を読み取らなくてはならない状況がしばしば生じる。オープンソースの開発でもやはり仕様書の作成は軽視されがちであるとされるが、インターネット上では、さらに、コミュニケーション手段が制限されるため、ソースコードによって仕様の理解が図られることが多いかもしれない。

したがって、情報処理活動という観点からオープンソースの開発におけるコラボレーションをとらえる場合、オープンソース自体が、ボランティア開発者の開発活動を媒介する働きを分析することは有益であると考えられる。ここでは、コラボレーションを特徴づける、コラボレーションの目標と、

コラボレーションに用いられる言葉およびメディアという3つの観点から分析を試みる。オープンソースの開発におけるコラボレーションの場合には、それぞれ、どんなソフトウェアをつくるかを規定する仕様と、コラボレーションのために用いられるプログラミング言語、既存の仕様を伝え新たな成果をつないでゆくためのソフトウェア自体が該当する。すなわち、開発するソフトウェアの仕様が多くの人々に理解されている場合や、採用したプログラミング言語が広く使われている場合、ソフトウェア自体が機能を追加したり修正したりしやすい構造になっている場合には、コラボレーションは生じやすいと考えられる。

まず、コラボレーションの目標であるソフトウェアの仕様が、コラボレーションにともなう情報処理の負荷を左右することが予想される。つまり、オープンソースの仕様が明確に規定され、コミュニティのメンバーに共有されているかどうか、コラボレーションのやりやすさに影響しうることを意味する。明確な仕様が共有されている場合、どのように開発するかについての多様な見解を統合し皆に伝える必要はないため、情報処理の負荷は低下する。Linux は、その機能が UNIX と重なる部分が多く、仕様を決めたり、それを共有したりする必要があまりなかったと考えられる。Raymond (1998) は、オープンソースのデザインが、目新しいものではなくすでに確立され慣れ親しまれたものであることの必要性を説いたが、情報処理活動という観点からは重要な指摘であろう。

OSASK 本体の開発と専用アプリケーションの開発とでは、仕様の理解についての違いは大きい。そもそも OSASK 本体は OS であり、その仕様は複雑で難しいうえに、基本的なコンセプトも、エミュレータ OS というかなりユニークな特徴を持っていた。また、仕様はあらかじめ規定されていたわけではなく、ほぼ手探りに近い形で開発が進められてきた。そのことは、OSASK が、ある程度まで開発された後で、大幅な設計の見直しが行われ、全面的に作り直されることが繰り返されてきたことから理解できる。こうした状況では、開発プロセスにともなう情報処理の負荷はきわめて大きいと言えよう。一方、専用アプリケーションについては、テキスト・エディタや電卓などが開発されているが、それらはいずれもシンプルなソフトウェアであり、その仕様について多様な解釈が生じることは少ないと思われる。それ以前に、OSASK の専用アプリケーションについては、ボランティア開発者が単独で開発しており、共同開発のために仕様の共通理解を図る必要すらなかった。

コラボレーションを行うための言葉であるプログラミング言語も、コラボレーションにともなう情報処理の負荷を規定すると考えられる。通常、仕様書などドキュメントの作成が軽視されがちなオープンソース開発においては、そのソースコードが仕様を理解するための重要な手がかりとなる。また、ソフトウェアについて何らかの問題を見出した場合、それを改善しようとすれば、それを実現するためのプログラムを作成し、オープンソースの作者に送付しなければならない。しかしながら、使用されているプログラミング言語が未知のものである場合、ソースコードを理解したり作成したりするためには、そのプログラミング言語を学ばなくてはならない。つまり、プログラミング言語が一般的なものでなく、普及率が低い場合には、コラボレーションを行うために多大な情報処理の負荷がかかることを意味する。

実際、OSASK の開発プロジェクトにおいては、プログラミング言語の影響が顕著に表れていると

考えられる。OSASK の専用アプリケーションの開発には一般的な C 言語を用いることが可能であるが、OSASK 本体の開発には、基本的には独自のアセンブリ言語 ASKA を使用しなければならない。このことが、OSASK 本体の開発に協力する開発者の数を減らしている可能性は高い。事実、コミュニティ・メンバーからも、一般的な C 言語での開発に切り替えることを求める声があがっている。

最後に、ボランティア開発者による追加開発や改善をつなぐためのソフトウェア自体のあり方も、コラボレーションにともなう情報処理の負荷に影響すると考えられる。ここでは、ソフトウェア自体のあり方を、ソフトウェア・アーキテクチャの統合度としてとらえる。ソフトウェア・アーキテクチャの統合度とは、すなわち、ソフトウェアが全体として最適化されている程度、を意味する。例えば、Linux の開発においては、ソースコードの品質や性能にやや問題があっても、有用な機能であれば、そのパッチ・プログラムを採用したことが知られている (Torvalds and Diamond 2001)。それはソフトウェアの最適化の状態を崩しうるものであるが、Linux においては、ある部分においては、最適化よりも機能性を重視する傾向があったと考えられる。

しかしながら、OSASK 本体は、効率性と高速処理を実現するべく最適化を図ることが、最も重視されている。OSASK も、保守効率などを理由に、独立性の確保された複数のサブシステム（事例ではモジュール）に分割されてはいるものの、サブシステムの使用関係をネットワーク化して再利用性を高め、無駄のない構造になるよう設計されている。また、プログラミング言語にアセンブリ言語を採用する理由も、高級言語を使用することで実現される最適化の程度では、満足できないからである。この場合、OSASK 本体の一部を修正したり、新たなサブシステムを追加したりすることは、最適化状態を崩すことになりうる。OSASK 本体に対して、部分的に開発協力をしようとしても、常に全体最適を考慮しながら行わなくてはならず、開発者には高い情報処理能力が要求されるのである。

すなわち、アーキテクチャの統合度が高いことは、ソフトウェアに追加開発や改善の余地ないし自由度が少ないことを意味し、結果として、ボランティア開発者が何らかの貢献を果たそうとしても、容易にはそれを受け付けられないという状況をもたらす。それは、オープンソースが、コラボレーションを媒介するメディアとして機能しえていないことを示すものである。OSASK の本体は、まさに、こうした状態にあるように思われる。

OSASK 本体の開発に対して、専用アプリケーションの開発は、ボランティア開発者が自由に行うことができ、最適化は要求されない。そのため、仕様を理解したり、最適化が実現されているかどうかを確かめたりする必要はない。したがって、専用アプリケーションの開発にともなう情報処理の負荷は低いと評価できる。

このようにオープンソースの開発におけるコラボレーションを制約する要因として、情報処理活動という側面から考察した。OSASK の本体の開発と専用アプリケーションの開発では、明らかに前者のほうが情報処理活動の負荷が高いと判断された。情報処理活動の負荷の違いは、OSASK 本体の開発と専用アプリケーションの開発におけるコラボレーションの成果の差に明確に表れていると考えられる。

3.3 分析結果

本稿においては、OSASKの開発プロジェクトを、そのコラボレーションを動機づける要因と制約する要因から分析してきた。ここでは、動機づけ要因として、社会政治的モチベーション、技術的モチベーション、経済的モチベーションをとりあげ検討した。まず、プログラミングをしたい、世の中の役に立ちたい、仲間からの名声を得たい、といった社会政治的モチベーションと、ソフトウェアの開発を成功させるためにオープンソースという手段を利用しようとする技術的モチベーションについては、OSASK本体の開発と専用アプリケーションの開発との間に顕著な差はないと判断した。ただ、現在あるいは将来に獲得される経済的な利益への期待に起因する経済的モチベーションの点では、独自のアセンブリ言語を使用しなければならないOSASK本体の開発は、一般的な言語を使用できる専用アプリケーションの開発よりも、将来のキャリアのために技術的なスキルを身に付けたいというメンバーに対する動機づけの力が弱いと考えられた。

また、コラボレーションを制約する要因については、オープンソース開発における情報処理活動という観点から分析を試み、コラボレーションに不可欠と思われる目標、言葉、メディアを考察した。オープンソースの開発は、仕様によっていかなるソフトウェアをつくるかという目標が設定され、プログラミング言語という言葉によって開発者の意図が理解あるいは表現され、公開されたソフトウェア自体をメディアとしてそこに機能が追加されたり改善されたりすることによって進められる。このとき、仕様が明確かつ広範に理解されているほど、また、採用されたプログラミング言語の普及率が高いほど、さらには、ソフトウェアのアーキテクチャに自由度が許容されるほど、必要とされる情報処理活動の負荷は小さく、ボランティア開発者はコラボレーションしやすいと考えられる。OSASK本体の開発においては、仕様を理解することが難しく、独自のプログラミング言語を採用し、アーキテクチャの自由度もないため、そのコラボレーションは、きわめて情報処理の負荷が高いと考えられる。分析の結果は、図10のようにまとめられる。

図10 OSASK開発における動機付け要因と制約要因の影響

		コラボレーションへの影響	
		OSASK 本体	専用アプリ ケーション
動機づけ要因	社会政治的モチベーション	○	○
	技術的モチベーション	○	○
	経済的モチベーション	△	○
制約要因	仕様の理解	×	○*
	プログラミング言語の普及率	×	○*
	アーキテクチャの自由度	×	○*

注) OSASKの専用アプリケーションは、小規模でありボランティア開発者によって単独で開発されたものが多い。*印は、当該の要因の影響を受けない可能性があるものを示す。

こうした分析結果から、OSASK 本体の開発は、OSASK の専用アプリケーションの開発よりも、動機づけの点で弱い部分があり、同時に、情報処理活動の負荷という点で強い制約が働いていると判断できる。したがって、OSASK の専用アプリケーションの開発には 20 名以上のボランティア開発者が貢献しているにもかかわらず、OSASK 本体の開発には、具体的な貢献がほとんどないという状況は、このような理由によって生じていると考えられる。

4 おわりに

本稿では、OSASK の開発事例をもとに、オープンソース開発プロジェクトの成否を左右する条件を検討してきた。OSASK プロジェクトにおいては、さまざまな理由から OSASK 本体の開発についてのコラボレーションが生じず、開発が遅れていることが問題となっていた。しかしながら、われわれは、OSASK の事例を、失敗の事例としてとりあげたつもりはない。オープンソースの開発プロジェクトの成否は、ボランティア開発者とのコラボレーションの成否によってのみ決まるわけではない。OSASK プロジェクトでは、ユーザーがさまざまなコミュニティ運営に協力しているし、専用アプリケーションの開発にも多数のボランティア開発者が貢献している。このような協力は、オープンソースという手段をとらなければ得られなかったものであろう。ただ、やはり OSASK 本体の開発において有効なコラボレーションが生じることは、コミュニティを活性化させ、OSASK の成長を加速するとも考えられる。本稿は、そうした方策をとるうえでの基礎的な手がかりを提供しうるのであると考える。

本稿の分析は、OSASK の開発プロジェクトの事例のみを対象に、OSASK 本体の開発におけるコラボレーションと、専用アプリケーションの開発におけるコラボレーションの違いを、動機づけ要因と制約要因という視点から相対的に説明するものであった。したがって、他の一般的なオープンソースの事例を対象とするものではなく、本稿で使用した枠組みが広く適用できるかどうかは明らかではない。それは、今後、OSASK の事例と他の事例とをひとつひとつ相対化し、枠組みの有効性を確認していかななくてはならない。もし、これらの要因において同等と評価されるのに、コラボレーションの状況が大きく異なるとすれば、他の要因を検討し特定しなければならないだろう。

また、動機づけ要因と制約要因としてとりあげたものは、必ずしもすべてを網羅したものではないし、相互に独立なものでもない。例えば、独自のプログラミング言語を使用することが、経済的モチベーションを下げ、開発者間のコラボレーションを制限してしまう可能性を指摘したが、こうした独自性は、一方で OSASK の存在意義の根拠となり、社会政治的モチベーションを強化しうる。また、仕様の共通理解の程度やプログラミング言語の普及率は、コラボレーションの成否を左右しうる重要な要因であるが、仕様が明確に規定され、普及率の高いプログラミング言語を採用したとしても、アーキテクチャの統合度が高い場合、コラボレーションを増大させることは難しいと思われる。なぜなら、高度に最適化が図られている場合、仕様を明確にしいかなるプログラミング言語を用いても、それを修正したり、新たな機能を追加したりすることは、やはり最適化を崩してしまう可能性があるか

らである。つまり、アーキテクチャの統合度は、オープンソース開発におけるコラボレーションを制限する要因として、より重要な意味を持つ可能性がある。しかしながら、このような変数間の詳細な関係については、本稿では明らかにされていない。

このような課題が残されているものの、本稿は、オープンソースの開発や、ソフトウェアのオープンソース化を成功させるための条件の一部を整理して示しえたと考える。特に、オープンソースの開発や改善といった活動を、ボランティアの開発者とのコラボレーションを通じて実現しようとする場合には、コラボレーションへの動機づけ要因と制約要因をうまくコントロールしなければならないことを指摘した。

しかしながら、動機づけ要因と制約要因は、必ずしも常にコントロールできるものではない。OSASKの開発プロジェクトにおいては、エミュレータ OS というコンセプトの実現を目標としているために、統合度の高いアーキテクチャとアセンブリ言語を採用せざるをえず、コラボレーションには不向きな状況が避けられず生じていた。このことは、ソフトウェアをオープンソース化すれば、ボランティア開発者とのコラボレーションが常に生じるわけではないことを示唆する。つまり、ソフトウェアには、その性質によって、ボランティア開発者とのコラボレーションを前提としたオープンソース開発に適当なものとそうでないものがあることを意味する。本稿の枠組みは、ボランティア開発者とのコラボレーションを狙ったオープンソース開発プロジェクトを計画する際に、当該のソフトウェアが、オープンソースという開発手段に向いているかどうかを判断する手がかりを提供しうる。

社会の基盤としての情報システムを制御するソフトウェアが、その内部を秘匿されたまま一部の企業によって提供されることは、ヒューマン・セキュリティ上のリスクとなりうる。オープンソースは、そうした問題を解決する可能性を持つ手段であるが、その開発プロジェクトを成功させることは難しい。本稿が、オープンソースの開発プロジェクトを成功に導くための条件を明らかにするための出発点となることができればと考える。

参考文献

- Brooks, Frederick Phillips, Jr., 1995, “The Mythical Man-month: Essays on Software Engineering,” Anniversary edition, Addison-Wesley. (滝沢徹・牧野裕子・富澤昇訳『人月の神話—狼人間を撃つ銀の弾はない』ピアソン・エデュケーション、1996年。)
- Feller, Joseph and Fitzgerald, Brian, 2002, “Understanding Open Source Software Development,” Peason Education.
- Raymond, Eric S., 1998, “The Cathedral and the Bazaar,” <http://www.tuxedo.org/~esr/writings/cathedral-bazaar/>. (山形浩生訳『伽藍とバザール』光芒社、1999年。)
- Simon, Herbert A., 1996, “The Sciences of the Artificial,” Third edition, The Massachusetts Institute of Technology. (稲葉元吉・吉原英樹訳『システムの科学 第3版』パーソナルメディア、1999年。)
- Torvalds, Linus and David Diamond, 2001, “Just for Fun: The Story of An Accidental Revolutionary,” Harper Collins Publishers. (風見潤・中島洋訳『それがぼくには楽しかったから 全世界を巻き込んだリナックス革命の真実』小学館プロダクション、2001年。)

Policy and Governance Working Papers*

Number	Author(s)	Title	Date
1	Tomoyuki Kojima and Mitsuaki Okabe	What is Policy and Governance Research?	November 2003
2 #	Michio Umegaki	Human Security: Some Conceptual Issues for Policy Research	November 2003
3	Takiko Fujii and Moriyuki Oe	Ageing and Generation Changes in the Tokyo Metropolitan Suburbs—A Study on Stable and Secure Habitation for the Aged—	November 2003
4	Soichiro Moridaira	Derivatives Contract for Event Risk	November 2003
5	Toshiyuki Kagawa and Akira Ichikawa	Natural Disaster and Governance of Regional Government: A Case of the 1997 Oder River Flood in Poland	December 2003
6	Wanglin Yan, and Aya Matsuzaki and Mikako Shigihara	Mapping the Spatial Structure of Regional Ecosystems and Calculating the Value of Trees for Regional Environment Governance with GIS Tools	December 2003
7	Hitoshi Hayami, Yoko Wake, Kanji Yoshioka, and Tomoyuki Kojima	The Possibility and Practice for CDM in Kangping Province in Shenyang: Policy Collaboration between Japan and China for Human Security	December 2003
8	Sayuri Shirai	European Monetary Union and Convergence in Monetary and Fiscal Policies—Human Security and Policy Response—	December 2003
9	Mitsuaki Okabe	International Financial Integration and the Framework of Economic Policy	December 2003
10	Masaaki Komai	The Integrated Evaluation of Price and Quality in Selecting PFI Contractors	December 2003

*Working papers marked with“ # ”are written in English, and unmarked papers are written in Japanese. All the papers are accessible on the internet homepage of the Center of Excellence (COE) program and can be downloaded in PDF file format (with some exceptions). A booklet version of the paper may be obtained by email at <coe2-sec@sfc.keio.ac.jp>. All the researchers affiliated with the COE Program are strongly encouraged to submit research papers to this working paper series.“Instructions to Contributors”are included at the end of the working paper as well on the home page of this COE Program <<http://coe21-policy.sfc.keio.ac.jp/>>.

11	Atsuyuki Kogure	Life Table and Nonparametric Regression: An Inquiry into Graduation of Standard Life Table for Japanese Life Insurance Companies	January 2004
12 #	Lynn Thiesmeyer	Human Insecurity and Development Policy in Asia:Land, Food, Work and HIV in Rural Communities in Thailand	January 2004
13	Satoshi Nakano Woojong Jung Xueping Wang	An Attempt towards the Multilateral Policy Collaboration for Human Security in Northeast Asia: Possibilities of CDM (Clean Development Mechanism) among Japan, China and South Korea	January 2004
14	Kanji Yoshioka Tomoyuki Kojima Satoshi Nakano Hitoshi Hayami Hikaru Sakuramoto Yoko Wake	The Practice of Tree Planting in Kangping Province in Shenyang: Policy Collaboration China for Human Security	February 2004
15 #	Yoshika Sekine, Zhi-Ming YANG for Human Security and Xue-Ping WANG	Air Quality Watch in Inland China	February 2004
16 #	Patcharawalai Wongboonsin	Human Security and Transnational Migration: The Case in Thailand	February 2004
17 #	Mitsuaki Okabe	The Financial System and Corporate Governance in Japan	February 2004
18 #	Isao Yanagimachi	Chaebol Reform and Corporate Governance in Korea	February 2004
19	Mikako Ogawa Masaki Umejima Jiro Kokuryo	RFID as Consumer Empowering Technology —Its Deployment in Japan—	February 2004
20	Mikihito Hayashi Jiro Kokuryo	Development of Open-source Software —Human Security through Disclosure of Key Technologies—	February 2004
21	Toru Sugihara Jiro Kokuryo	Creating a New Method Measuring Capability of University Students	February 2004

Instructions to Contributors

Revised February 25, 2004

1. (The purpose of the series) The working paper series, covering researches conducted under the 21st Century Center of Excellence (COE) Program “Policy Innovation Initiative: Human Security Research in Japan and Asia,” aims at timely publication of research results and eliciting comments and furthering debate. Accordingly, all the researchers affiliated with the COE program (twenty nine members whose names are listed on the COE web page) are strongly encouraged to submit relevant research papers to this series. The nature of the COE program is explained on the homepage (see the URL at end of this note) .

2. (The nature of the papers) The series includes research papers written, as a general rule, in Japanese, English, or Chinese language. Given the aim of the series, the papers of the series include reports of ongoing research, papers presented at the COE-sponsored workshops and conferences, relevant published research papers, as well as original unpublished formalized research papers. Although the papers may vary in their theme and scope, all papers are expected to address either the issue of policy and governance or its methodology, or the issues involved in the various aspects of human security. Specifically, the relevancy to the issue should be expressed in the title or subtitle of the paper, or in the abstract of the paper.

3. (Submission procedure) Contributors are requested to store the paper in a single document file (using, as a general rule, MS Word or LaTeX) and to transmit the paper as an e-mail attachment. It should be sent to the editors of “Policy and Governance Working Paper Series” (see below for the e-mail address) . Hard-copy printouts of the manuscript are not required unless editors specifically request them. Working papers are going to be continuously published and there is no time limit for submission.

4. (The requirement of the author) While the COE members and Keio University Fujisawa-Campus researchers may submit papers directly, all other collaborating researchers are requested to submit the paper to one of the COE members who are expected to edit, correct and ensure that it meets the criteria of the series.

5. (Refereeing) Given the aim of the series, there is no refereeing process per se. However, any submitted paper may be excluded, if the editorial committee regards the manuscript inappropriate for the series. The editorial committee may ask for minimum revisions before printing. Upon acceptance of the paper, the Secretariat may request the author to provide original data (such as Photoshop EPS) to improve the clarity of the printing.

6. (Submission fee) There is no submission fee. Twenty copies of the paper will be provided to the author free of charge (and more will be available upon request) .

7. (Copyright) Copyright for all papers remain with the authors.

8. (Forms of publication) All the papers are made accessible in two ways: (a) in a booklet form, and (b) in downloadable PDF file format on the internet homepage of this COE program.

9. (Style instructions) Although all the papers will be reformatted before printing, authors are requested to make the manuscripts conform to the following format:

1) The manuscript should be typed with double line-spacing (including all notes and references) on A4 size paper.

2) The font size should be 10.5-11point in the case of Japanese or Chinese, and 11-12 point in the case of English. (In the case of other languages than these three, interpret the guidelines appropriately here, and below also.)

3) The title page (page 1) should contain the following information: (1) the title; (2) the name(s) and affiliation of the author(s), (3) the email addresses of the author(s), (4) the background of the paper, such as conference presentation, and acknowledgments (if applicable) . If the paper is in any way funded by the COE or its related programs, it must be so mentioned.

4) The second page is for the abstract of the paper. The abstract must be in a single paragraph that summarizes the main argument or the conclusion of the paper in about 150 words in the case of English, and 7-12 lines of characters in the case of Japanese or Chinese. At the end of the abstract, a list of four to six keywords should be included. If the paper is written in languages other than Japanese or English, a corresponding Japanese or English version of the abstract should also be printed.

5) Main text should begin on page 3. Beginning from the cover page (page 1) , all pages should be numbered consecutively.

6) Footnotes should be numbered consecutively and should be placed at the bottom of the appropriate page.

7) Tables and charts may (1) be placed in the appropriate place in the text, or (2) be prepared on separate pages and attached at the end of the text, provided that the place to be inserted is indicated in the text.

8) Reference list must be attached at the end of the text. Only works referred to in the text should be included in the list.

9) Although there is no limit of the length of the paper, the editorial committee requests that the paper be of approximately 10-30 pages in length.

10. (The revision of the instructions) This Instructions to Contributors will be revised from time to time, and the current version is always shown on the COE web page.

11. (Correspondence)

-Submission of the paper: coe2-wp@sfc.keio.ac.jp

-Requesting the booklet version: coe2-sec@sfc.keio.ac.jp

-PDF file version of the paper: <http://coe21-policy.sfc.keio.ac.jp/>

Editorial Committee Members of the Working Paper Series:

Mitsuaki Okabe (Managing Editor), Michio Umegaki, Masaaki Komai.