慶應義塾大学

夢のＯＳ、ＯＳＡＳＫ[1]

－オープンソースによるソフトウェア開発－

OSASK 計画代表の川合秀実氏は、先月までの懸念事項であった資金面での問題を克服し、ひとしきりの安堵感を覚えていた。川合氏がこの 2 年半、寝食以外のほとんどの時間を費やして開発に取り組んできたオペレーティング・システム（以下、OS）「OSASK」（おさすく）が、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、その開発援助資金として年間約 300 万円が支給されることになったのである。

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、ソースコードが公開され、インターネット上で無償配布されているといういわゆる「オープンソース・ソフトウェア」（以下、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こそが OSASK 計画であり、そのビジョンに共感した人々が自発的に協力参加するオンライン・コミュニティによって運営されていた。

これまで川合氏は、収入を得るためのアルバイトなどは一切行なわず、学生時代に奨学金を節約してためた貯金を食いつぶしながら、OSASK の開発に専念してきた。それは、お世辞にも速いとはいえない開発状況を案じた OSASK コミュニティのメンバーから「OSASK の開発に集中してほしい」と念を押されていたからであった。OSASK 開発の中心的役割を担っていた川合氏が、開発に専念できなくなることは、これまで、ただでさえ遅かった開発速度をますます停滞させてしまうことを意味したのである。しかしながら、その貯金も次第に底をつき始め、「そろそろアルバイトを始めざるをえない」と思い始めていた。今回の資金援助は、そんな矢先の吉報に他ならなかった。

しかし、川合氏は開発に専念できることの喜びを噛み締めながらも、今後いかに開発を進めていくかについて頭を悩ませていた。OSASK 計画の最大の問題は、その開発に多大な時間がかかっていることであり、ゴールが見えないばかりか短期的な計画さえも十分に守られていない点にあった。その主な原因の一つは、OSASK の最も際立った特徴である「効率性」を実現するために、開発言語として独自のアセンブリ言語「ASKA」を利用していることにあった。事実、コミュニティ・メンバーからも、開発速度を上げるために、開発言語として C 言語を採用することが提案されていた。しかし、C 言語によって開発を行なえば、現状の OSASK の効率性はあきらめざるを得ない。川合氏は、これまで通り開発を進めていくか、それとも、C 言語による開発に切り替えるか、決断しなければならなかった。

---

[1] 本ケースは、慶應義塾大学大学院経営管理研究科國領二郎教授の指導のもと、同研究科博士課程の林幹人が作成した。ケースは、経営管理に関わる討議用の教材として開発されたものであり、特定の組織や個人に関する経営管理の巧拙を例示するものではない。なお、本ケースの著作権は慶應義塾大学に所属する。（2003 年 3 月 26 日）

■ＯＳＡＳＫとは

1998 年こそ、「オープンソース」という概念を、世界中に知らしめた年であった。まず、1 月には、ネットスケープ社が、同社の主力製品である WWW ブラウザ Netscape Communicator をオープンソースとして公開した。9 月には、オープンソースの代名詞とも言うべき Linux のディストリビューション[2]を販売するレッドハット社に、インテル社とネットスケープ社が出資することが報じられた。世界有数の一流企業が、オープンソース企業に出資するという事実もさることながら、インテル社は、ソフトウェアをクローズドにする戦略をとって成功を収めてきたマイクロソフト社と、「Windows」と「Intel」を合成して「Wintel」と呼ばれるほどの同盟関係を築いていただけに、その対抗勢力とでも言うべきオープンソース企業への出資は人々を驚かせた。そのニュースとの関連は明らかではないが、その直後の 10 月には、マイクロソフト社がオープンソースを脅威と見なしていることを裏付ける内部資料の流出事件があり、オープンソースの勢いをさらに知らしめることとなった。また、「フリーソフト」という名称[3]が、本来「自由なソフトウェア」という意味であるにもかかわらず、「無料のソフトウェア」という誤解を招いてきたことを受け、「オープンソース」と呼びかえられたのも 1998 年であった。この名称変更によって、オープンソースがビジネスと矛盾しないものであることが理解されるようになり、その後の普及を加速させたと言われている。

OSASK 計画が開始されたのは、奇しくもオープンソースが一躍脚光を浴びたこの年のことであった。「OSASK」とは、OS＋ASK から作られた造語であり、短く、他にはない文字列で、ロゴのバランスの良さと、ひらがなで書くとかわいいから、といった理由で川合氏が名づけたものであった。OSASK は、当初はまだ、オープンソースではなかった。

ＯＳＡＳＫのユニークなコンセプト

OSASK は、「エミュレータ OS」というコンセプトを掲げる OS であり、エミュレータ機能を重視する OS である。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 上に Machintosh の画面を再現し、その中で Machintosh 用のアプリケーションを動作させるようなソフトウェアである。

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。このことはユーザーにとっては不便であり、その結果、皆が使っているからという理由で、欲しくもない OS を選択しなければならないという状況が生じる。既存 OS のこうした状況に

---

[2] ディストリビューションとは、オープンソースとそのアプリケーション組み合わせてパッケージ化したものである。

[3] 必ずしも厳密なものではないが、「無料のソフトウェア」という意味を持つフリーウェアと、フリーソフトとは区別される。

不満を覚えた川合氏は、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有効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。つまり、OSASK は、Windows や Machintosh、UNIX といった OS をエミュレーションすることによって、それらの OS 用に開発されたアプリケーションを利用することができるように企図された OS に他ならない。

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、現在普及している OS と同様に、ハードウェア資源の割り当てやデータの管理を行い、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、他のアプリケーションと同じく、OSASK の OS 部分の上で動作するものであり、例えば、Windows エミュレータや Unix エミュレータなど、エミュレーションしようとする他の OS にあわせて開発されるものである。ただ、エミュレータ部分に関しては、OS 部分と一体化することで、利用者がその構造を意識することなく、既存の OS と同じように使用できるよう考慮されていた。

ここで、エミュレータを動作させる際に問題になるのはスピードである。川合氏は、大手企業が販売する既存の OS が、極めて非効率な構造をしており、そのためスピードが遅く、不安定でさえあると考えていた。そして、そういった OS の上でエミュレータを動作させても、スピードの点で問題が生じるのは目に見えており、効率性の問題を解決しない限り、エミュレータ OS の構想を実現することは難しいと考えた。そこで、川合氏は、極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。

こうして、互換性と効率性という二つを柱に、OSASK の開発が企画された。既存の OS への問題意識から生まれたそのコンセプトは、まさにこれまでの OS の常識を覆すものであり、現在普及している OS に不満を持った人々から支持を得ていた。

■オープンソースＯＳＡＳＫが誕生するまで

ＯＳＡＳＫ計画代表 川合秀実氏

小学 4 年生のとき、叔父から FM－8 というお古の 8 ビット・マシンを譲り受けたことが、川合氏とコンピュータとの出会いであった。FM－8 は、当時でさえ既に旧式のマシンであり、メモリ 64KB、1 メガヘルツ・クロックという今では考えられないスペックであった。そのため、コンピュータ・ショップでも FM－8 用のソフトウェアは販売しておらず、仕方なく、マシンに付属していたプログラム言語の BASIC を独学で学び、簡単なプログラムを作成して遊んでいた。

当時、川合氏の友人は 16 ビット・マシンを持っていたが、川合氏の FM－8 との処理能力の差は大きかった。また、そのころ流行していた「任天堂 ファミリー・コンピュータ」も 8 ビットであったことから、「同じ 8 ビットなのに、どうしてこんなに処理速度が違うんだ」と考え始めた。そして、試行錯誤を重ね、コンピュータやソフトウェアの仕組みについて知識を深めた結果、マシンの性能を最大限引き出すために、アセンブラでプログラミングすることを覚えたのである。

中学 2 年生になり、川合氏は、FM－8 に限界を感じ、その上位機種の FM－7 を中古で手に入れた。スペックは上がったものの、やはり当時としては既に旧式の機種であった。

川合氏は、プログラミングが上達するうちに、「こういう命令があったらいいな」と、BASIC を改造し始めた。そのうち、「起動のシーケンスが気に入らない」とか、「メモリを増やそう」といった具合に、次第に、コンピュータの性能を最大限に引き出すための、より内部的な処理のコントロールに興味を持ち始めた。そしてこの頃、OS も作れるものであることを知り、OS を設計し始めたのであった。しかし、当時はまだ、コンピュータの知識が十分とは言えず、プログラミング技術もそれほど高くはなかったことから、OS は完成しないまま時は過ぎていった。

フリーウェア「Ｖ９８」の開発と「川合堂」の設立

1994 年に横浜市立大学文理学部に進学した川合氏は、FUJITSU の FM－TOWNS を購入した。そして、FM－TOWNS 上で NEC の PC9801 用のソフトウェアを動作させるためのエミュレータ「V98」の開発を行なった。1996 年になり、V98 の完成度がようやく満足できるレベルに達したため、川合氏は、これを公開し普及させたいと考えたが、当時、すでに FM－TOWNS の最盛期は過ぎつつあり、普及は難しくなっていた。

そこで川合氏は、1996 年 4 月、V98 をはじめ、自らや友人たちが開発したソフトウェアの普及支援を行なう組織「川合堂」を、友人と共に設立、川合氏はその代表に就任した。同年 8 月には、V98 をフリーウェアとして公開するに至った。V98 は当時、フリーウェアであったものの、ソースコードは公開されておらず、オープンソースではなかった。しかし、V98 の反響は川合氏の予想をはるかに超え、「こんなに喜んでもらえるとは思わなかった」というほどのものであったが、このとき川合氏はフリーウェアの楽しさを知った。

川合堂は、その後、FM－TOWNS の衰退と共に、1998 年から開始された OSASK の開発と普及を主に支援する組織となっていった。

ＯＳＡＳＫ計画の萌芽

川合氏は、中学生の頃から OS の設計に挑戦していたものの、開発能力が未熟だったこともあり、実際のソフトウェアとして実を結ぶことはなかった。しかし、大学生になった川合氏は、V98 の開発などを経て、ソフトウェア開発のための技術を身に付けていった。そして、大学院に進学した 1998 年、より開発の実現可能性が高まったことを感じた川合氏は、OSASK の開発に着手したのである。翌 1999 年には、OSASK 開発に関するメーリング・リスト「OSASK－ML」を立ち上げた。

しかしながら、大学院在学中の 2 年間は、OSASK の開発が大きく進展することはなかった。それは、川合氏が、自らを「勉強好き」と評するように、学業の物理学の研究にのめり込んでしまったためであった。

2000 年 4 月、大学院を修了した川合氏は、家族の反対を押し切り、OSASK の開発に専念することを決めた。そして、2000 年 3 月 31 日、OSASK 開発のメーリング・リストに、開発再開宣言（参考資料１）を投稿した。ただ、開発再開宣言といっても、ある程度まで完成した開発途上のソフトウェアがあったわけではなく、OSASK のビジョン以外は、まったくのゼロからのスタートといってよかった。

オープンソースまつり

2000 年 12 月、川合氏は、エミュレータ「V98」の評判を受けて、「オープンソースまつり 2001 in 秋葉原」というイベントのパネラーとして招待された。このイベントのエミュレータ・セクションで行なわれるパネル・ディスカッションは、当時、オープンソースやフリーウェアのエミュレータ開発に取り組んでいた著名な開発者数名によって行なわれることになっており、川合氏はその中の一人であった。川合氏は、V98 と、OSASK 両方の紹介をすることになっていた。

2001 年 2 月 10 日のイベント当日、川合氏がスピーチするために与えられた時間は約 10 分であったが、FM－TOWNS が下火になっていたこともあって、そのほとんどを OSASK の説明にあてた。川合氏は、会場から非常に良い反響を感じ取っていたが、その感覚は間違いではなく、イベント終了後に、ある雑誌記者から取材を申し込まれたほどであった。

フリーウェアからオープンソースへ

当初は、V98、OSASK とも、フリーウェアであったが、オープンソースではなかった。しかしながら、「オープンソースまつり」への参加をきっかけに、川合氏は急きょ、V98 と OSASK をオープンソースにすることを決めた。それは、「オープンソースまつりにパネラーとして出席するのに、自らのソフトがオープンソースじゃないのはまずいだろう」と考えたからであった。

オープンソースやフリーソフトのライセンスには、GNU GPL や FreeBSD コピーライトが採用されることが多いが、OSASK のライセンスは、「川合堂ライセンス」という独自のパブリック・ドメイン・ソフトウェア（PDS）ライセンスである。川合堂ライセンスは、V98 と OSASK のオープンソース化に合わせて、「オープンソースまつり」に間に合わせる形で作成され、2000 年 12 月 30 日に公開された。（参考資料２）

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とするものであり、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、事実上の著作権放棄のためのライセンスである。日本の法律上、著作権を放棄することはできないが、川合氏は、このライセンスを作ることによって著作権の放棄と同等の状況を作ろうとしたのであった。GNU GPL や FreeBSD コピーライトは、フリーと言いながらも、何らかの制約を設けていたため、川合氏は「完全に自由にしたい」という理由からこ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、受けは良かったものの、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。

■これまでのＯＳＡＳＫ開発

川合氏は、OSASK の「開発再開宣言」で宣言したとおり、インストールの容易なドキュメント付きの一般公開版を、毎月リリースしてきた。ときには、OSASK 自体にほとんど変

化が見られないこともあったが、それでもリリースは欠かさなかった。（参考資料３）

開発とリリースに関する川合氏の基本的な考え方は、「完成度よりも、とにかく動くものを作る」ということであった。実際、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、起動と同時に同じアプリケーションが三つ起動するだけのプログラムを開発した。

こうして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単なる技術的なテスト・プログラムでしかなかった。しかも、それ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、ソースコードの公開された真のオープンソースになるきかっけとなった「オープンソースまつり」までには、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 3.0 に達した現在、いまだ OS としては十分な機能を提供できていないものの、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。（参考資料４、参考資料５）

これまでは、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。

ＯＳＡＳＫのモジュール化

OSASK の開発は、これまで、ある程度開発しては大幅に作り直し、また、ある程度進歩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なわれた。それまで、OSASK のコア部分は、モノリシック[4]な構造をしていたため、これをインタフェースの規定されたモジュール単位に分割したのである。

モノリシックな構造をしたプログラムは、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。モジュール化によって、そ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で完結させることができるため、メンテナンスの負荷を下げることができる。またそのことは同時に、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

[4] モノリシックとは、ここでは一つのプログラムですべての機能を実現しようとする非モジュールの状態を意味している。

共同開発できることを意味する。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なうためのものであった。

OSASK の内部は、14 のカテゴリに分類され（参考資料６）、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。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（参考資料７）。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りえた。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まったく独立に修正することが可能であった。

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、下位の関数は、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、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は得られていると実感していたものの、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。そうした状況から川合氏は、モジュール化によって得られている最大のメリットは、ソースコードの中から問題箇所を特定しやすくすることにあると考えていた。

しかしながら、インクリメンタルに開発が進められるオープンソースにとって、モジュール化を厳密に行なうことはきわめて難しい。つまり、開発初期においては、必ずしも最終的な完成図を描けないのであり、初期段階で分割されたモジュールによって、後に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。特に、ユニークなコンセプトを持つ OSASK にとっては、それはさらに困難なことであった。そして、ソースコードの修正が、モジュールの中だけで終わらないとき、言い換えれば、その修正がインタフェース部分にまで及ぶとき、多大な修正作業が必要となる。そのような場合には、膨大な数のモジュールをチェックし、呼び出し部分を見つけ出して修正しなければならなかった。

OSASK の開発において、ある程度開発しては繰り返し行なわれてきた大幅なリフレッシュは、このモジュールのインタフェース部分の修正と関係していた。しかしながら、そのような見直しの頻度も少なくなり、次第に安定的に開発が行なわれるようになってきた。

開発言語ＡＳＫＡ

OSASK は、そのサイズや処理速度などの効率性を最も重視して企画された 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、C 言語や BASIC、Java などの高級言語に対して、低級言語と呼ばれる言語である。それは、高級言語のように人間の言葉に近い体系をしているものではなく、コンピュータが直接理解できる機械語を、人間が理解しやすい記号に単純に置き換えただけのものである。また、ハードウェアへの依存度が強く、その内部構造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なうえ、汎用性も低い。

しかし、アセンブリ言語を使用することで、効率の良いソースコードを書くことが可能となる。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る必要がある。ここで、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。仮に、ある高級言語で書かれた「2＊3」という一つの処理命令が、2 と 3 を掛けることを表しているとすると、「掛ける」という操作のないコンピュータの内部では、例えば「2 をメモリに書き込む、メモリの値 2 と 2 を足してメモリに書き込む、メモリの値 4 と 2 を足してメモリに書き込む」などといった手順で処理される。つまり、高級言語での「2＊3」という一つの処理命令は、「メモリに書き込む」「メモリの値を取り出す」「足す」「繰り返す」といった複数の機械語命令の組み合わせによって実現されているのである。ここで、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といったソフトウェアによって機械的に翻訳されるが、その結果生成される機械語は多くの場合、冗長で無駄の多いものとなる。これに対してアセンブリ言語の命令は、「2＊3」といった便利な書き方ができない代わりに、「メモリに書き込む」「メモリの値を取り出す」「足す」「繰り返す」といった機械語の命令に対応した機能範囲の小さな命令が存在し、それを組み合わせることで、無駄のない、効率的なソースコードを書くことができるのである。事実、OSASK のサイズは、現在のバージョン 3.1 で約 50KB であり、既存の OS とは比較にならないほどの効率性を実現していた。

ただ、高級言語であればひとつの命令で済むところを、アセンブリ言語では、機械語に合わせて一つ一つ書いてゆく必要があるために、開発速度は低下する。ある報告によれば、同じ処理のプログラムを書く場合、アセンブリ言語は、行単位で C 言語の 2 倍を超える量のソースコードを書かなければならないという。また、アセンブリ言語は、ハードウェアに強く依存しているため汎用性が低く、その習得者は C 言語よりもはるかに少ない。そのため、ソフトウェア開発にアセンブリ言語を使うことは、開発に時間がかかる要因となりえた。

そこで川合氏は、ASKA という C 言語ライクなアセンブリ言語を開発し、OSASK の開発に使用した。C 言語ライクと言っても、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（参考資料８）。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。

こうして、OSASK はその高い効率性を実現するために、ASKA によって開発されていた

が、C 言語で開発することも可能であった。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いたため、特に理由がない限り ASKA を使用することにしていた。

結果として、OSASK の開発に参加するためには、アセンブラやハードウェアの知識があり、川合氏が開発した ASKA を習得するという条件を満たす必要があった。そのため、誰もが容易に開発に参加できるというわけではなく、開発メンバーの増員は簡単ではなかった。事実、オープンソースという形で開発しているにも関わらず、主要な開発参加者が 5 名にとどまっている状況が、そのことを物語っていた。

しかし、それだからこそ、OSASK 計画のビジョンに強く共感でき、かつ、技術知識の高い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、開発メンバーの中には「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」とまで言う参加者さえいた。

一方で、OSASK 専用のアプリケーションについては、ASKA を利用する必要はなかったため、開発参加のハードルはそれほど高くなかった。その結果、アプリケーションの開発には、開発経験の豊富な上級者からプログラムを勉強中の初心者まで、幅広い開発者が参加していると川合氏は考えていた。

開発メンバーのプロフィール

OSASK コミュニティのメンバーは、OSASK 本体の開発に 5 名、OSASK の上で動作するアプリケーションの開発に 20 名ほどの参加者があたっていた。また、ユーザーやテスターとして協力しているメンバーが 100 名から 200 名ほどいた。

中心的なメンバーの数はこれまで、「オープンソースまつり」に出たり、雑誌で取り上げら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はなかった。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」と感じていたものの、やはりメンバーの数は十分ではなく、実際に、OSASK 開発で手伝って欲しい案件について「求人」という形でホームページに掲載していた（参考資料９）。

OSASK コミュニティでは、これまでに一度もオフ会[5]を開催したことがなかった。川合氏も、オフ会の実施については、これまで何度も考えたものの、特に資金面の理由から断念せざるを得なかったのである。そのため、川合氏は、川合堂メンバー以外の OSASK コミュニティのメンバーとは、直接会ったことがなかった。

しかし、OSASK を通じてやり取りするなかで、コミュニティのメンバーが、およそ 10 代後半から 30 代中盤くらいまでの、主に学生や職業プログラマによって構成されていると、川合氏は考えていた。メンバーの技術レベルも様々であったが、OSASK 本体の開発に携わる中心的なメンバーのレベルは一様に高いと感じていた。ただ、川合氏は、OSASK のビジョンに賛同してもらえる人であれば誰でも参加して欲しいと考えており、その人がどこの誰でどのような経歴や技術を持っているかについて詮索することはなかった。

---

[5] オフ会とは、電子ネットワーク上で知り合ったメンバーなどが、実際に集まって親睦を深めるための会合のことである。オンラインに対して、オフラインでの会合を意味する。

川合氏とメンバーの関係

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、個人的なメールと、始めたばかりのチャットで行なわれていた。主に、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をとっていた。パッチ・プログラムなどを送付してくるメンバーは少なく、このことは、OSASK 本体のソースコードの約 8 割が川合氏によって書かれているという現状からも明らかであった。

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た。そのため、開発メンバーからのパッチ・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ていた。パッチ・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、送付されてきたソースコードをレビューし、問題のある箇所などを修正して、OSASK に組み込んだ。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、バグや効率性、スピードなどの観点から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、川合氏が納得するまで議論が行なわれた。そして、川合氏自身が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った。

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し、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考えがあったからであった。そして、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ていた。そして川合氏は言うのであった。「お互いに競争しようよ」と。

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いた。問合せには、例えば「OSASK って何ですか？」「メモリレス・アーキテクチャとは何？」といった質問から始まり、「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、「いつ頃できるんですか？」「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」と言うものまであった。

OSASK に関する問合せは多数にのぼり、内容も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認

識していた。そのため、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。また、個々の問合せにも、川合氏は、一件づつ丹念に対応し、理解を促そうと心がけた。

また、川合氏は、そのように対応することで、OSASK コミュニティの理解をも促し、いずれ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、新規の問合せには、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った。そのおかげで、川合氏は、問合せ対応の負荷は、徐々にではあるが確実に減少していると感じていた。

■ＯＳＡＳＫ開発の将来

ＯＳＡＳＫの開発予想

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いた（参考資料 10）。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ほとんど自信がなかったし、OSASK の完成時期に至ってはまったくわからなかった。このことは、ロードマップが、「開発計画」ではなく、OSASK の「開発予想」と名づけられていることに如実に表れていた。

川合氏は、開発が計画的に進まない理由は、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、通常、ユーザーや共同開発者とのやり取りの中で、その方向性が決められ、全体のイメージがつくりあげられていく。つまり、オープンソースの完成図は、その開発過程において次第に具体化され、ユーザーや共同開発者から予期せず提示される追加的なアイデアによって、書き換えられていくのである。インクリメンタルに開発されるオープンソースに関しては、昨日の完成図が今日のそれとは異なることを意味していた。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。そして、明確な全体像や完成図がないために、どうしても開発の過程で構造上の問題や限界にぶつかり、大幅なリフレッシュが必要になると考えられた。

このようにオープンソースの開発においては、開発計画を立てることは容易ではない。しかし、たとえ計画どおりでなくとも、ソフトウェアの日々の顕著な成長が、開発メンバーやユーザー協力者などをモチベートすることが知られていた。ところが、現在の遅い開発スピードは、完成イメージを描くことを難しくしているばかりか、人々を不安にさえさせていた。そしてその不安がさらに、OSASK の実現可能性に関する問合せを増やしたり、潜在的な協力者の参加を難しくしたりすることによって、開発速度の向上を阻害しているように見えた。

開発方針の再検討

OSASK の開発速度を上げるためには、大きく二つの方向があると考えられた。一つは、

問合せの対応やコミュニティの運営などの雑務を誰かに引き継ぎ、OSASK 計画における第一の開発者である川合氏の時間のすべてを、開発作業にだけ投入しようとするものである。実際、問合せに関しては、過去に同様の問合せ対応がなされている場合には、コミュニティのメンバーによって自発的に、川合氏に代わって回答されるようになってきていた。しかし、OSASK のコンセプトは極めてユニークかつ難解であるため、川合氏以外にその全貌を把握しているメンバーがいないという状況があり、完全に引き継ぎを行うことは、現在のところ困難であった。

もう一つは、川合氏以外の開発能力を増強する方針が考えられた。現状では OSASK 本体の 8 割以上を川合氏が開発していたが、これを多くの開発者に分散させようとするものである。これまで開発参加者が増加しなかった原因は、OSASK の完成イメージの共有が難しいことと、独自のアセンブリ言語 ASKA を採用していることにあると考えられた。ただ、前者については、OSASK の難解なコンセプトこそ、OSASK のアイデンティティであるためこれを変えることはできず、問合せへの対応などを通じて地道に理解を促してゆくしかないと思われた。結果として、取りうる選択肢として考えられるのは、OSASK の開発言語の見直し以外にはなかった。

実際、開発言語については、遅々とした開発状況に一抹の懸念を抱いた OSASK の開発メンバーからも、「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、その機能やグラフィカル・ユーザ・インタフェース（GUI）などを、まず、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。利用者の多い C 言語の使用と、イメージの共有によって、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っていた。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」を体現したものに他ならなかったからである。

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方法の中で、最良のものであると認識していた。しかしながら、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は、やはり難しい選択であった。これまで通りアセンブリ言語で開発を続けていくべきか、あるいは、C 言語を中心とした開発に切り替えるべきか、川合氏は、選択に迫られていた。

■エピローグ

川合氏は、OSASK 計画存続のための資金の問題もクリアされ、初めてのオフ会を開催したいと考えていた。日ごろから OSASK 開発に協力してもらっているメンバーをねぎらうと

同時に、皆で集まって OSASK の将来について語り合いたいと考えたのである。川合氏は、周囲にプログラマ仲間のいなかった自らの小中学校時代を振り返り、できるだけ若いメンバーにオフ会への参加機会を提供したかった。そして、遠方にいる若いメンバーには、交通費の一部を負担してあげようとも考えていた。

2002 年 10 月 17 日、今日も川合氏は、OSASK の開発場所と化した自宅の一室で、川合氏自作の開発機に向かっていた。そして、まだ見ぬオンライン上の同志に思いを馳せながら、静かにキーボードをたたき始めた。

こんにちは、川合です。
いきなりですが本題です。かねてからあちこちで話題になった
OSASK **に関するオフ会をやりましょう！という話題です。**
・・・

●参考資料１：ＯＳＡＳＫの開発再開宣言

[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/

注：OSASK のメーリング・リストに投稿された川合氏の開発再開宣言（一部省略)。

●参考資料２：川合堂ライセンス

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も責任はとれません。」ってことです。

利用者や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し、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には事前の著作者の許可が必要だか、特許をとること以外については何ら制限はない。

３．補足
　基本的に、コピーは大歓迎です。もし、著作者に何か恩を感じたら、一人でも多くの人にこのソフトウェアをすすめて、コピーしてあげて下さい。著作者は多くの人に使ってもらいたいと思っているので、コピーすれば著作者は喜びます。それでも足りないと感じたら、是非、感想を著作者に送ってあげて下さい。そうすれば、もっと喜ぶでしょう。
　これでライセンスされたソフトウェアの著作権情報だけを改変し、それを再配布することはこのライセンスによって禁止されていません。派生物扱いです。これは抜け穴ではないです。したがって、著作権情報だけを書き換えて再配布することが必要なら、していただいてかまいません。
　バグを取ったり、機能を追加していただくのはもちろんですが、プログラムに註釈を付けて読みやすくしたり、ドキュメントを補足するなど、そういうバイナリーや実行結果に反映されないような改変も大歓迎です。そういう派生物ができたら、連絡してもらえるとうれしいです（義務ではありません）。
　何か疑問点があったり、派生物を作成する上で情報が不足していると感じたら、著作者に連絡をとって質問することができます。ただ、著作者の都合ですぐには返事ができないかもしれません。それはご容赦ください。このライセンスそのものの不備などを指摘する場合は、著作者か川合堂にご連絡ください。
　もしかすると、これでライセンスされたソフトウェアで利用されている技術について、著作者の許可の無い者が特許を取得することを禁じていることが、日本の特許法に抵触しているかもしれません（ご意見を待っています）。これについて、ライセンスを最初に提唱した川合秀実の見解を以下に書いておきます。
　特許法が制定された背景には、発明者が自分の発明による利益を守るために発明の詳細を公表しないことが科学技術の進歩を遅らせるから、公表してもらう代わりに一定期間の独占利用を法的に保護する、という精神があります。このライセンスでは、発明の詳細を意図的に隠すつもりはなく、したがって特許を取ることを禁止しても特許法の精神には反していないと考えています。むしろ、このライセンスの精神を汲まない者が特許を取得しオリジナルのソフトウェアや派生物に対して特許料を請求するかもしれない不安の方が、科学技術の進歩を遅らせると考えます。したがって、その不安を事前に払拭したこのライセンスは特許法に適ったものだと考えています。
　もちろん、一番安心なのは、オリジナルのソフトウェアで発明と認められるすべてのものについて著作者が特許を取得して、他者の特許取得を事前に妨げればいいのですが、それは著作者には負担になる場合があります。その負担とソフトウェアを公開するかどうかを天秤にかけなければいけないとしたら、公開をあきらめてしまう場合もあるかもしれません。それは特許法の精神の期待するものではありませんし、我々の目的（「１．目的」を参照）にも沿いません。
　また、発明の詳細が特許法の形式によって記述されていないために明らかでない場合、著作者以外のものが許可なくそれを解析して明らかにすることは、このライセンスで認められていています。それを無償で公開してもいいですし、有償で販売することもできます。
　なお、これでライセンスされたソフトウェア中のすべての技術に対して、利用に際して特許料を支払う心配が全く無いわけではありません。公開の時点で有効な特許による技術がソフトウェア内で使われていれば、それについては特許保持者からの特許料要求がありえます。このライセンスが特許料の心配無しと保証しているのは、このソフトウェア内で新たに発明と認められる技術に対してのみです。
　このライセンスを自分のソフトウェアに適用したいと思う方がおられましたら、事前・事後に許可を求めることなく使っていただいてかまいません。もし不都合があれば、ライセンス文を改変して使っていただいてもいいです。改変の際には、混乱を防ぐためにライセンス名を変更するのを忘れないようにしてください。

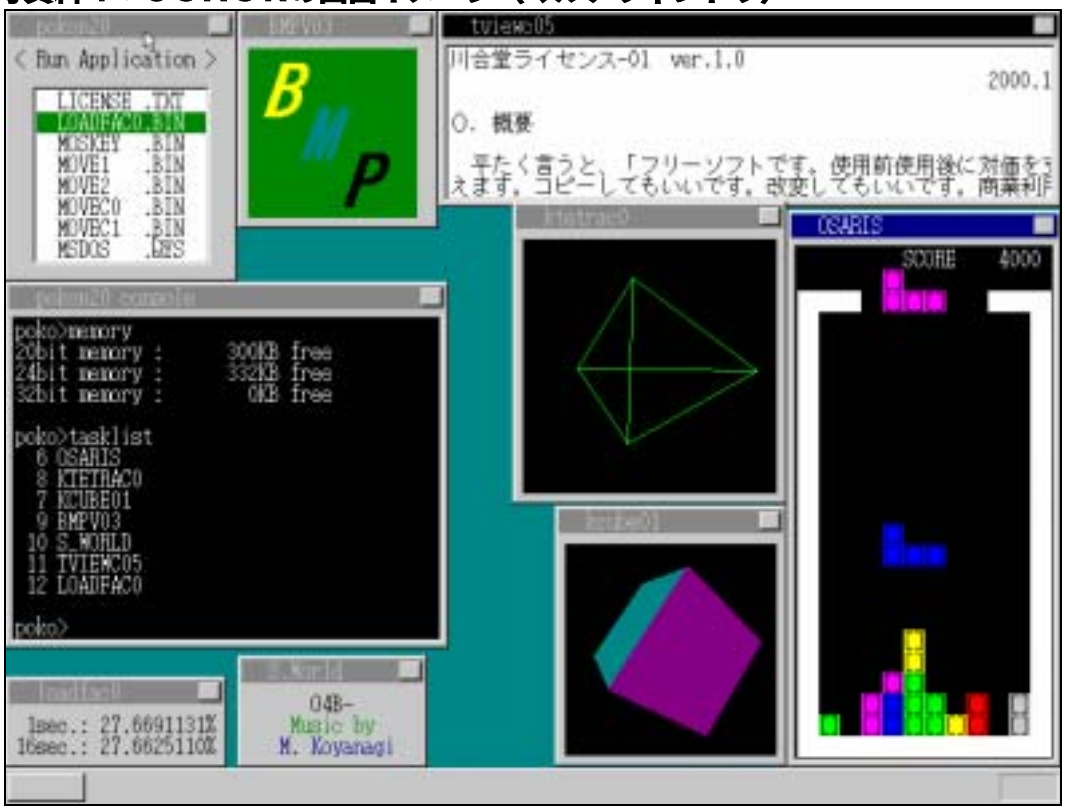
出典：http://www.imasy.org/~mone/kawaido/license01-1.0.html

●参考資料３：ＯＳＡＳＫ発展の軌跡

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）。
2001/07/07	OSASK/AT ver.1.4, OSASK/TOWNS ver.1.4	えせ仮想記憶を搭載。カラーフォントサポート。 OS のイメージファイルを圧縮して格納することで、起動速度がさらに向上。
2001/08/01	OSASK/AT ver.1.5, OSASK/TOWNS ver.1.5	えせファイルシステムを搭載(リードオンリー)。
2001/09/01	OSASK/AT ver.1.6, OSASK/TOWNS ver.1.6	アプリケーションの改定。 DLL のサポート。
2001/10/01	OSASK/AT ver.1.7, OSASK/TOWNS ver.1.7	グラフィック API の整備。バンドルアプリケーション大幅追加。 TOWNS 版ではセミハイレゾ(768x512)に対応。
2001/11/02	OSASK/AT ver.1.8, OSASK/TOWNS ver.1.8	日本語表示のサポート(暫定仕様)。アプリケーションの改定。
2001/12/01	OSASK/AT ver.1.9, OSASK/TOWNS ver.1.9	文字表示が正式仕様に。韓国語表示に対応。 フォントなどのデーター圧縮率が向上。 pokon の改良。仮想画面に対応(TOWNS 版)。
2002/01/06	OSASK ver.2.0	ファイルの書き換えが可能になった(しかし実メディアにはライトバックできない)。 pokon が簡易的な拡張子の関連付けに対応。仮想画面に対応(AT 版)。
2002/02/08	OSASK ver.2.1	ファイルの書き換えの実メディアへのライトバックが可能になった。些細な改良とバンドルアプリケーション追加。
2002/03/01	OSASK ver.2.2	些細な改良とバンドルアプリケーション改定。
2002/04/01	OSASK ver.2.3	tek0 圧縮ファイルの読み込みに対応。バンドルアプリケーション改定(へらへらアニメプレイヤ)。
2002/05/08	OSASK ver.2.4	バンドルアプリケーション改定(MML プレイヤ、スクリプトインタプリタ)。 NASK 開発中のために OSASK の開発は少ない。
2002/06/09	OSASK ver.2.5	壁紙表示対応。 TownsMENU 風デザイン対応。 NASK 開発中のために OSASK の開発は少ない。
2002/07/05	OSASK ver.2.6	画面デザインがさらに充実(超漢字風、NWSOS 風)。バンドルアプリケーション改定。 NASK 開発中のために OSASK の開発は少ない。
2002/08/01	OSASK ver.2.7	画面デザインがさらに充実(Win3.1 風)。 NASK リリース(OSASK ソースも NASK に完全対応)。

出典：http://www.imasy.org/~kawai/osask/oldver.html より作成。

●参考資料４：ＯＳＡＳＫの画面イメージ（マルチウィンドウ）



出典：http://www.imasy.org/~kawai/osask/picat20.html

●参考資料５：ＯＳＡＳＫ概要の説明（ＡＴ版バージョン3.0）

AT 版 version 3.0 (OSASK/AT version 3.0)

●免責事項

このプログラムはきっと安全です。動作チェックも綿密にやっていますが、川合秀実個人も、OSASK 開発グループも法的にはその動作を保証できません。実行の際は、各自の責任の範囲で行ってください。

自分がパソコンの初心者だと思う方は、詳しい人に手伝ってもらってください。

著作権は放棄していませんが、転載やコピーは歓迎いたします。詳しくは、添付の license.txt をご覧下さい。

●対応機種

実行できる機種は、i386 以上の CPU と 4MB 以上のメモリを搭載した IBM PC/AT 互換機です。そのほかの機種で実行しないでください。

ただし、AT 互換機であっても、必ずしもうまく動くとは限りません。あしからずご了承ください。

●おもな特徴

できること

- フリー（自由）でオープンソース。
- 起動が速い（直接起動ディスクによる起動の場合）。
- OS 自身が小さい（圧縮して 50KB ほど）。
- アプリケーションも（他の OS の同じようなものよりも）小さい。

- まっとうなマルチタスク、とりあえずマルチウィンドウ。
- 起動後は MS-DOS や MS-Windows を利用せず、ほぼ完全に 32bit で動く。
- ファイル読み込みと書き込み、メモリ管理が少しできる。
- 2000JIS 対応（第3水準まで）、韓国語対応（KS C 5601）。
- 解像度 800x600 にも対応。

あなたのパソコンは、こんなに小さなプログラムでも、これだけのことができるんだということを実感してください！

まだできないこと

- エミュレーションはできていません。
- VESA ファンクションを使わない高解像度切替えができません。
- USB 機器が全く使えません。
- フロッピー以外の外部記憶装置が使えません。
- ディスク ID を使った同一ディスクの識別ができません。
- JIS 第4水準は使えません。

VESA に対応していない一部のビデオカードでは、残念ながら 640x480 の表示しかできません。

HD 上にインストールされている OS やファイルに影響しません。ご安心ください。

前のバージョンとの違い

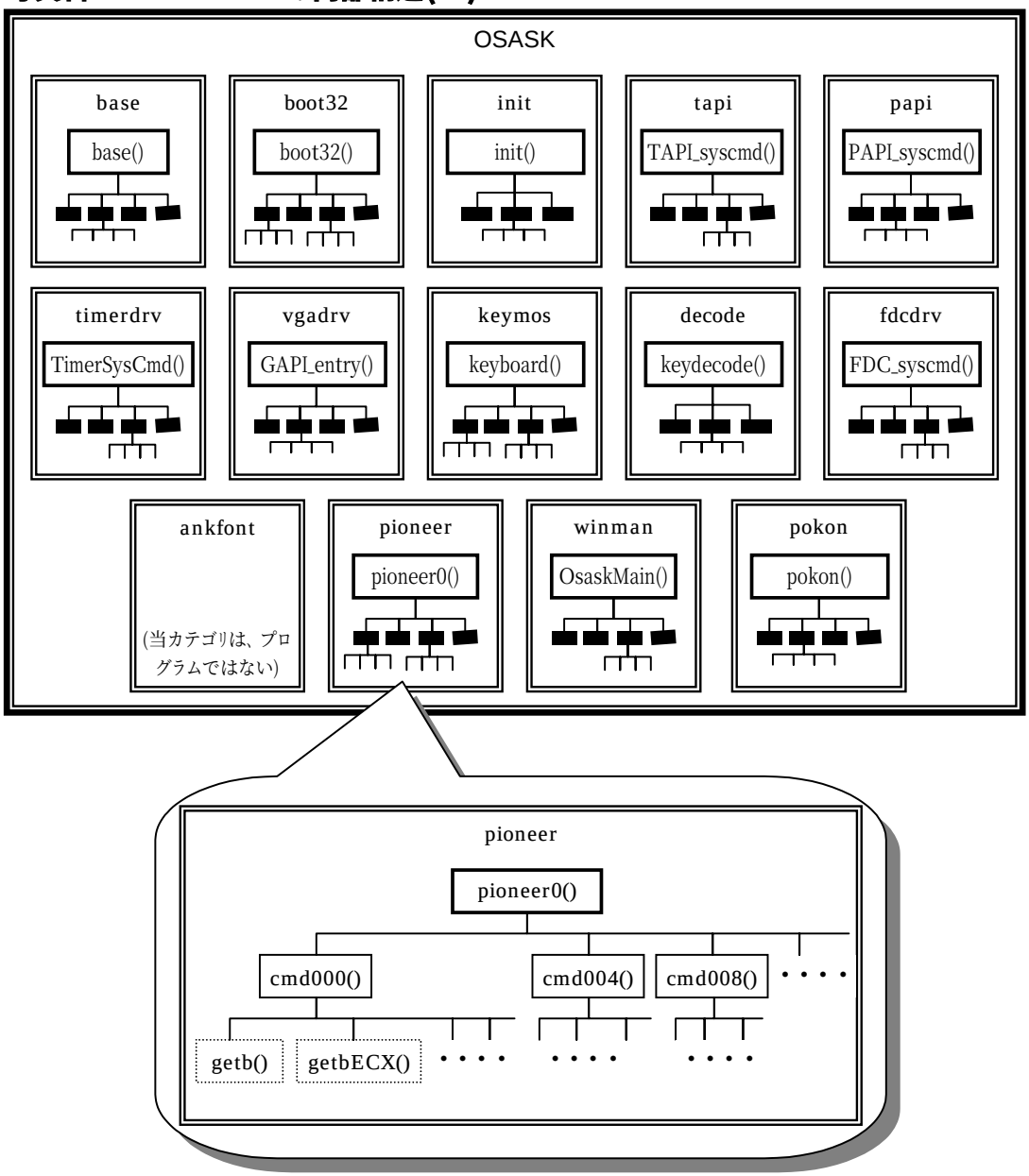
- インストールした OSASK がファイルとして見えるようになった。
- FDD のないパソコンでも起動できるようになった。
- 付属アプリケーションの変更。

出典：http://www.imasy.org/~kawai/osask/download.html から抜粋。

●参考資料６：ＯＳＡＳＫの内部構造(１)

名称	概要
base	機種依存のある起動処理
boot32	機種依存の少ない起動処理
init	雑用
tapi	タスクスケジューリング
papi	仮想記憶制御
timerdrv	システムタイマーのドライバ
vgadrv	ビデオドライバ(vesa32 や vesa16 や vesa8 もある)
keymos	キーボードとマウスのドライバ
decode	キーコードデコーダ
fdcdrv	フロッピーディスクのドライバ
ankfont	半角フォント
pioneer	API
winman	ウィンドウマネージャ
pokon	ファイルマネージャ

●参考資料７：ＯＳＡＳＫの内部構造(２)



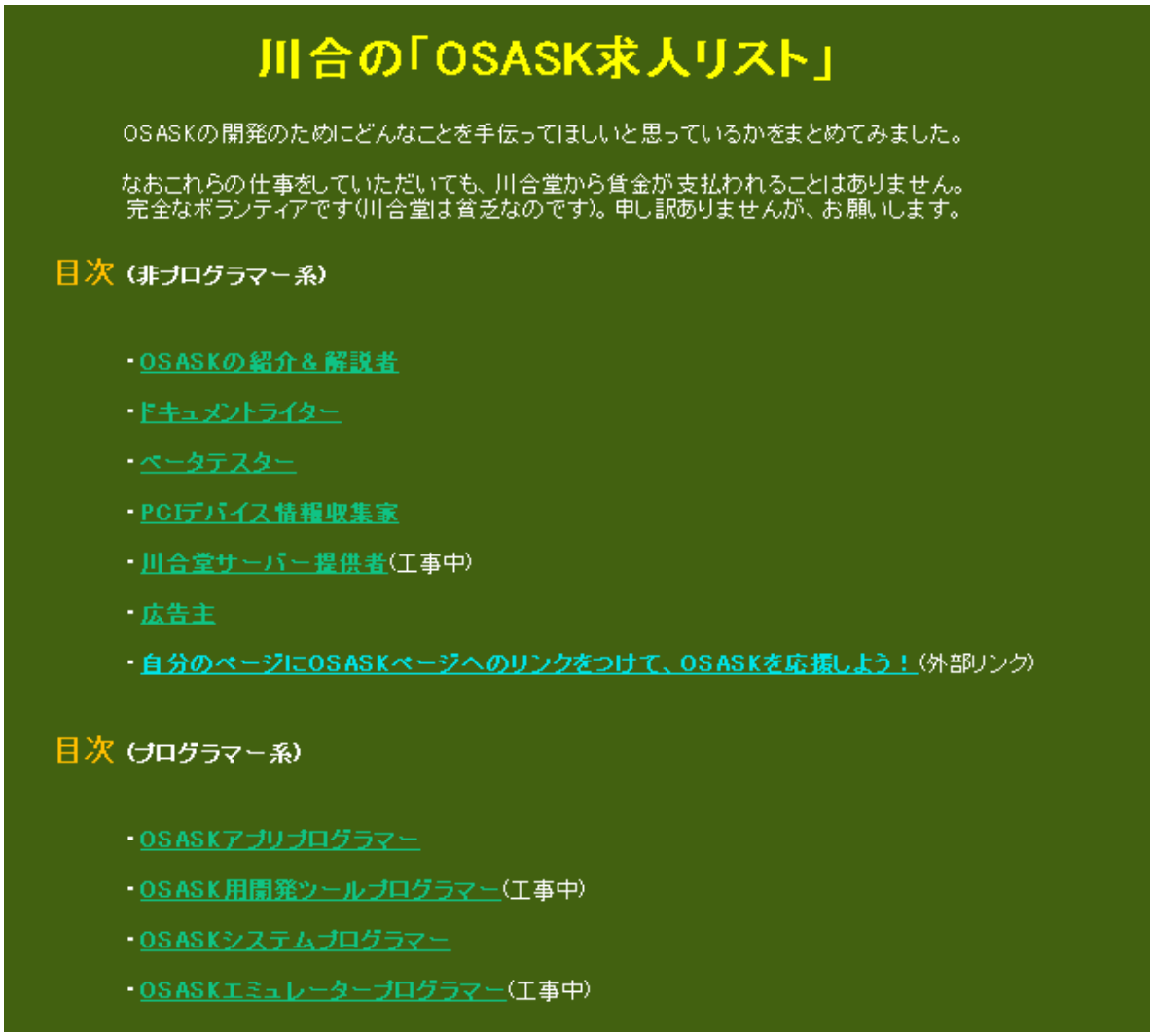
●参考資料８：一般的なアセンブリ言語とＡＳＫＡの違い

一般的なアセンブリ言語	ASKA（川合氏の開発したアセンブリ言語）

XOR EAX,EAX
XOR ECX,ECX
LABEL:
ADD EAX,ECX
INC ECX
CMP ECX,10
JLE LABEL
	int sum == EAX, i == ECX;
	sum = 0;
	i = 0;
	do {
	sum += i;
	i++;
	} while (i <= 10);

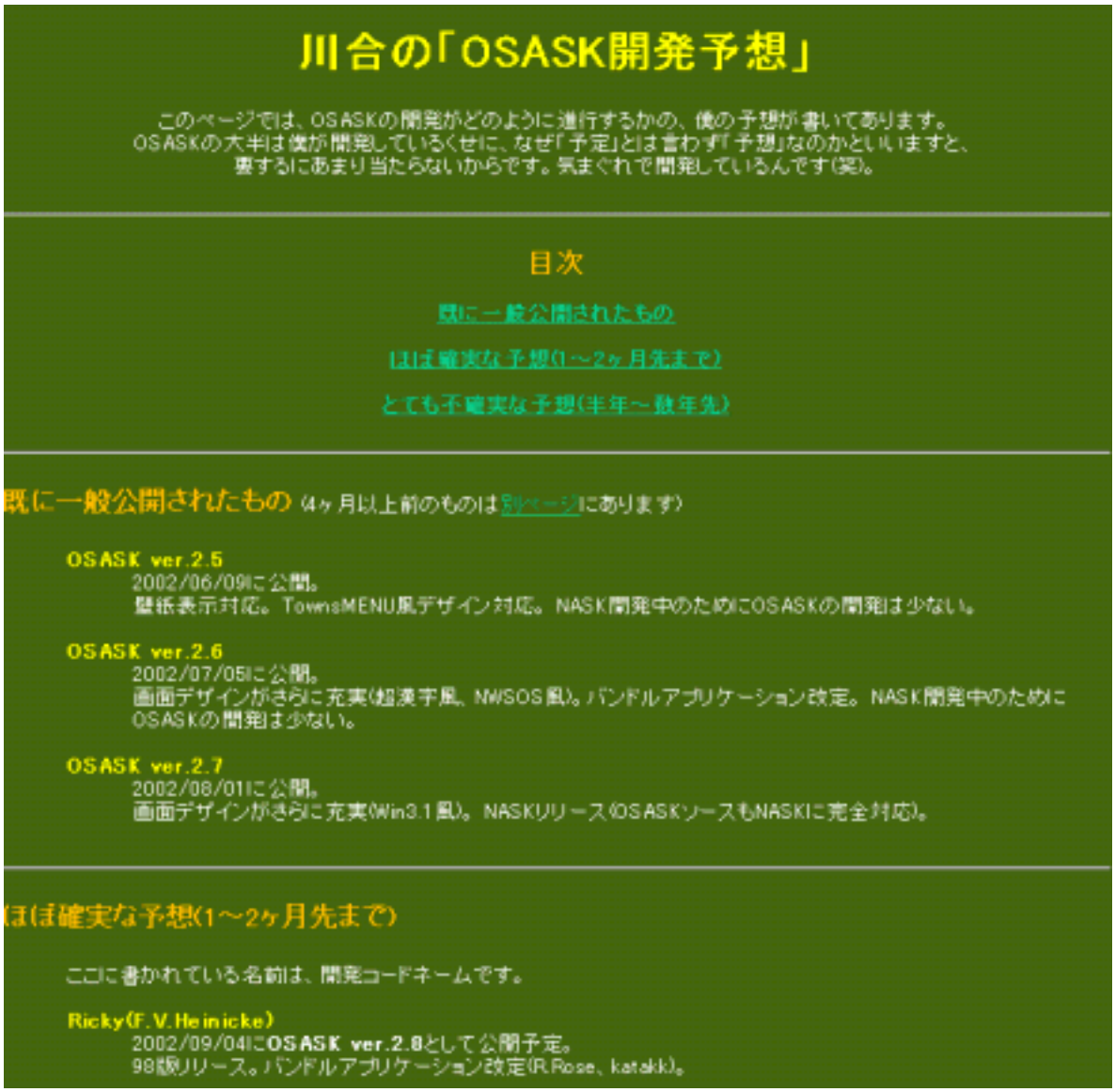
（上記の両方のソースコードは、完全に同じ機械語を出力する）

●参考資料９：ＯＳＡＳＫ計画の求人リスト



出典：http://www.imasy.org/~kawai/osask/kyujin.html から抜粋。

●参考資料 10：ＯＳＡＳＫ開発のロードマップ



川合の「OSASK開発予想」

このページでは、OSASKの開発がどのように進行するかの、僕の予想が書いてあります。
OSASKの大半は僕が開発しているくせに、なぜ「予定」とは言わず「予想」なのかといいますと、
要するにあまり当たらないからです。気まぐれで開発しているんです(笑)。

目次

既に一般公開されたもの

ほぼ確実な予想(1～2ヶ月先まで)

とても不確実な予想(半年～数年先)

既に一般公開されたもの (4ヶ月以上前のものは別ページにあります)

OSASK ver.2.5
2002/06/09に公開。
壁紙表示対応。TownsMENU風デザイン対応。NASK開発中のためにOSASKの開発は少ない。

OSASK ver.2.6
2002/07/05に公開。
画面デザインがさらに充実(超漢字風、NWSOS風)。バンドルアプリケーション改定。NASK開発中のために
OSASKの開発は少ない。

OSASK ver.2.7
2002/08/01に公開。
画面デザインがさらに充実(Win3.1風)。NASKリリースのOSASKソースもNASKに完全対応。

ほぼ確実な予想(1～2ヶ月先まで)

ここに書かれている名前は、開発コードネームです。

Ricky(F.V.Heinicke)
2002/09/04に**OSASK ver.2.8**として公開予定。
98版リリース。バンドルアプリケーション改定(R.Rose、katakk)。

出典：http://www.imasy.org/~kawai/osask/plan.html より抜粋。