OSS メッセージペディアと OSS エコシステム

OSS Message Pedia in OSS Eco System

前　原　志　好

要　約　Linux カーネルから出力されるエラー・警告メッセージに関するマニュアルは存在しない．本稿では，それらのメッセージについて，解説と対処方法が書かれたオンラインマニュアルの製作と，そのマニュアルが OSS エコシステムの中で果たす役割について触れる．次に，集合知の力と多くの目による監視で質の高いマニュアル作りを目指すため，Linux カーネル開発コミュニティへ働きかけ，開発者自身にもメッセージのドキュメント化作業の一部に取り組んでもらうための呼びかけについても紹介する．また，ユーザコミュニティへのアプローチとして，OSS メッセージペディアが用意しているサービスについて言及する．最後に，この活動に関する考察と，これからの活動方針について述べる．

Abstract　We have never seen any manual of Linux kernel messages. OSS Message Pedia is an online manual for error, warning and informational messages originated in Open Source Software, and contains descriptive texts and actions to be taken for these messages. This paper discusses this manual's role as a part of OSS eco system. The goal is to produce a high-quality manual cooperating with the wisdom of crows. To achieve success, we approach the Linux kernel developers about documentation of messages, and build the web site by name of http://ossmpedia.org/ for the Linux user community. In the end of this paper, I describe a part of the results of these activities that is in progress and the list to do in future.

1. は　じ　め　に

Linux® カーネル（以下，Linux）を含めたオープンソースソフトウェア（以下，OSS）は，ミッションクリティカル領域でも使われ始めている．様々な人や企業の努力により，商用の製品に見劣りしない機能や性能を備えてきた．しかし，OSS を利用する上で必要なドキュメントは，十分とはいえない．

一方，ミッションクリティカル領域で現在でも使用されているメインフレームの OS では，以下のようなドキュメントが存在する．

- 運用管理
- ジョブ管理コマンド解説
- システムコール
- エラー・通知メッセージ
- 新機能解説
- バージョン間非互換情報
- 導入ガイド
- 技術解説

Linux では，システムコールやシステム管理用ツールなどについては，man コマンドによ

るオンラインマニュアルが充実している．また，新機能・技術解説はカーネルのソースコードの Documentation ディレクトリにいくつか存在する．さらに，LDP[*1]（The Linux Document Project）というボランティアによって運営されているサイトもある．ここには，how-to からガイドまで様々なドキュメントがある．日本語のドキュメントでは，Linux JF プロジェクト[*2]が有名である．ドキュメントの豊富さで比較するとメインフレームと見劣りしない．

しかし，「エラー・通知メッセージ（以下，メッセージ）」，「バージョン間非互換情報（以下，非互換）」については，まとまったドキュメントはない．Unisys® 製メインフレーム ClearPath® Server CMP CS シリーズ・HMP® IX シリーズの OS である EXEC では，それらのドキュメントも新しいバージョンのリリースとともに提供されてきた．例えば，「メッセージ」に関しては，表示される全てのメッセージの解説が記載されている“HMP IX シリーズ・シリーズ 2200 EXEC 操作解説書メッセージ編 レベル 46”[1]などがある．「非互換」では，表示されるメッセージのカラムが一文字変更になったという情報や，システムコールのある条件でのエラーステータスの変更情報，削除された OS の生成パラメータの情報など，あらゆる非互換について記載されている．

本稿では，「メッセージ」マニュアルを補完するために開発した“OSS メッセージペディア（http://ossmpedia.org）”とそれが果たす役割について解説する．「非互換」については，本稿では扱わないが，非互換を検出する「Linux カーネル互換性テストツール（crackerjack[*3]）」が同時期に開発されているので，そちらを参照されたい．

2. OSS メッセージペディアとは

本章では，OSS メッセージペディアを作成した背景とそのシステムの構築内容について説明する．

2.1 プロジェクトの概要

ここでは，OSS メッセージペディアという Web サイトを作るきっかけとなったプロジェクトについて説明する．このプロジェクトは，独立行政法人 情報処理推進機構（以下，IPA[*4]）の 2006 年度オープンソースソフトウェア活用基盤整備事業 テーマ型（開発）「Linux メッセージ・マニュアル・データベースの作成」への公募案件としてスタートした．

2.1.1 公募の内容

公募では，Linux にメインフレームで提供されているようなエラーや警告メッセージに関するマニュアルが存在しないことを問題として掲げ，メッセージ情報をマニュアルとしてデータベース化し，随時，参照・更新できるシステムの構築を求めていた．このシステムを開発することにより，Linux 利用者は，障害発生時にメッセージを検索し，その意味を理解し，対処方法に従い障害を除去することができる．このシステムがない場合，出力されたメッセージの内容だけで障害を理解できればよいが，理解できない場合には，ソースコードから該当のメッセージを調査しなければならない．このコストは非常に高く，緊急な障害発生時にはすぐに対応できない．

その他の要件として，システムの構築には，全てのコンポーネントに OSS を使用するという項目もあった．

2.1.2 応募の背景

Linux はユニアデックス株式会社（以下，ユニアデックス）においてもメインフレーム，Windows® に次ぐ第三のプラットフォームとして注力している分野であり，企業としても Linux コミュニティに貢献したいという思いがあった．また，メインフレームとのギャップを埋めていくことが，ミッションクリティカル領域での Linux の活用につながるという考えに賛同し，この案件への応募を決断した．その結果として，ユニアデックスの提案が採用され，開発に着手することになった．

2.2 基本方針

以下を，システム構築の基本方針として挙げた．

1） メッセージのテキストイメージを得るためソースコードをスキャンする
2） ID として，メッセージ出力箇所のソースコードのパス名と行番号を使用する
3） 独自に構築する部分以外は全て既存の OSS を利用する
4） オープンな技術仕様（RSS，Atom[*5]，JSON，XHTML，YAML，HTTP，RSS Auto-discovery，Microformats[*6]）を積極的に使う
5） 国際化に対応する

1），2）に関して，補足する．メッセージは，出力イメージからではなくソースコードに記述されているテキストを使用し，メッセージの調査前にあらかじめ独自の ID を振る．ただし，この方針には，以下の問題を含んでいる．

- Linux バージョンが異なればパス名や行番号が変更になり，ソースコード上で同じ意味でも，別 ID となる
- 複数行で構築される一メッセージに対して複数の ID が割り当てられる

これらに対しては，Linux カーネルの中で ID を発番する以外には，根本の解決に至らない．そのため，まずは，上記の基本方針に従ってデータベースの設計を行い，ID が発番されれば，相互変換ができるような仕組みを取り入れることにした．OSS メッセージペディアが割り振った ID は，ソースコード上のメッセージ発行箇所を特定する ID であり，メッセージの内容を特定する ID ではない．真の ID を割り振るには，Linux カーネルコミュニティの協力を得る必要がある．ここは，5.1 節にて詳細に見ていく．

2.3 システム構成

次に，本システムの構成と，その構成要素について説明する．

2.3.1 全体図

OSS メッセージペディアは，図 1 の大きな点線で囲まれた部分で構成される．今回は，メッセージ・マニュアル・データベースの設計と，二重線で囲まれたメッセージを抽出する機能（スキャナ），Web インターフェースやデータベースアクセスを提供するアプリケーション（Web アプリケーション）の開発を行った．

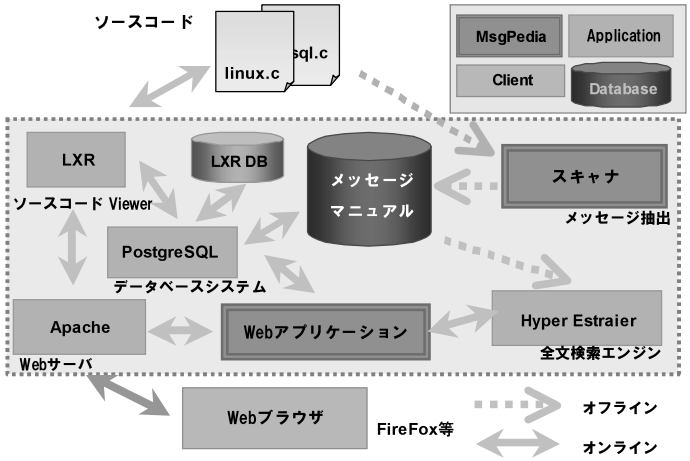

図 1　OSS メッセージペディアシステム構成図

2. 3. 2　OSS 部品

Web サーバには，実績の高い Apache，データベースサーバには，開発メンバーのスキルセットにマッチした PostgreSQL を採用した．また，メッセージを検索するためのエンジンとして，Hyper Estraier[*7] を採用した．PostgreSQL から利用するため，Hyper Estraier のコア API を使用して開発されている pgestraier も導入した．これは，SQL に pgest 関数を通して，全文検索のデータにアクセスしたり，PostgreSQL のあるカラムが更新されたことをトリガにして，全文検索のインデックスを更新するのに使用している．

ソースコードのビューワには，Web インターフェースでソースコードが閲覧できる LXR[*8]（Linux Cross-Reference）を採用した．最新の LXR は sourceforge.net® にホスティング拠点を移し，一から作り直されている．新 LXR は，リレーショナル・データベースにデータを保存する機能を持ち，C 言語以外で書かれたソースコードにも対応している．

2. 3. 3　開発部品

独自開発したアプリケーションのソースコードスキャナと Web アプリケーションについて補足する．

1）　ソースコードスキャナ

ソースコードから，メッセージの出力箇所を検索し，ファイル名，行番号，メッセージのテキストイメージなどを抽出し，ID の発番を行う．また，メッセージ出力の関数やマクロ（たとえば，ext3_debug など）を指定して，メッセージの抽出と ID の発番も行える．

2）　Web アプリケーション

Catalyst[*9] という MVC アーキテクチャスタイルの Web アプリケーションフレームワークを採用し，Perl 言語でアプリケーションを記述した．Catalyst は，ソフトウェアの更新が頻繁に行われ，プラグインも多数作成されており，VOX[*10] で採用されているなど，非常に活発なコミュニティを形成している．

また，Catalyst は，その名前のとおり，これだけで全てを解決するようなフレームワークではなく，Model や View の部分は別のモジュールに処理を委譲し，自分自身は非常に薄いレイヤーとなり，既存の資産を生かす設計となっている．

上記の開発部品やデータベースのスキーマは，sourceforge.net のソースコードリポジトリで，オープンソースとして公開[*11] している．

3. OSS コミュニティの仕組み

OSS は単に開発者がコードを提供するだけではなく，その周辺に OSS の利用者や支援者がいてはじめて成立する．本章では，それらの関係やマニュアル作成がどのような作業であるかについて見ていく．

3.1 OSS エコシステム

生態系と聞いて，まず自然界の生態系を思い浮かべるだろう．この自然界の生態系について，WikiPedia® の文を引用する．

> 生態系は大きく，生産者，消費者，分解者に区分される．植物（生産者）が太陽光から系にエネルギーを取り込み，これを動物などが利用していく（消費者）．遺体や排泄物などは主に微生物によって利用され，さらにこれを食べる生物が存在する（分解者）．これらの過程を通じて生産者が取り込んだエネルギーは消費されていき，生物体を構成していた物質は無機化されていく．それらは再び植物や微生物を起点に食物連鎖に取り込まれる．これを物質循環という．
> *「生態系」（2007 年 9 月 15 日 11：48 UTC）『ウィキペディア日本語版』.* [*12]

これを，OSS の世界に置き換えると，表 1 のように解釈できる．OSS メッセージペディアは，「OSS を支援する個人・団体」として分解者を演じる．役割は，ドキュメントを充実させることによって，OSS のエコサイクルを支援することである．

表 1　OSS エコシステム登場人物

役割	対象	説明
開発者（生産者）	OSS 開発コミュニティ	ソフトウェアを開発する．ボランティアや企業の社員として開発している．
利用者（消費者）	OSS 利用者	OSS を利用してシステムを構築する．自ら利用したり，そのシステムに対して対価を得たりする．
支援者（分解者）	OSS を支援する個人・団体	OSS をサポートする．使用した結果をレポートしたり，ドキュメントを作成したりする．開発者をバックアップする．

3.2 OSS エコサイクル

エコサイクルとは，自然界の食物連鎖と少し似ている部分もある．例えば，生産者が開発したソフトウェアを，消費者が利用して，利用した結果を分解者が生産者にフィードバックするという仕組みである．ただし，食物連鎖は捕食する側を上位に，被食される側を下位に置くピラミッド型で表現されることがあるが，OSS の連鎖では，捕食・被食の関係ではなく，それぞれが対等で提供・利用の関係である．

図 2 に，OSS エコサイクルが提供・利用の関係になっている図を示す．吹き出しに，それぞれの登場人物が提供されるものを記入した．時計回りのサイクルでは，利用者は支援者に安心の対価を払い，支援者は開発者に開発資金援助を行い，開発者は消費者に新機能を提供するという流れが見える．逆方向では，利用者は開発者に開発の推進力を与え，開発者は支援者が手がけるシステム構築のソリューションを提供し，支援者は利用者のシステム構築を支援するという流れが見える．

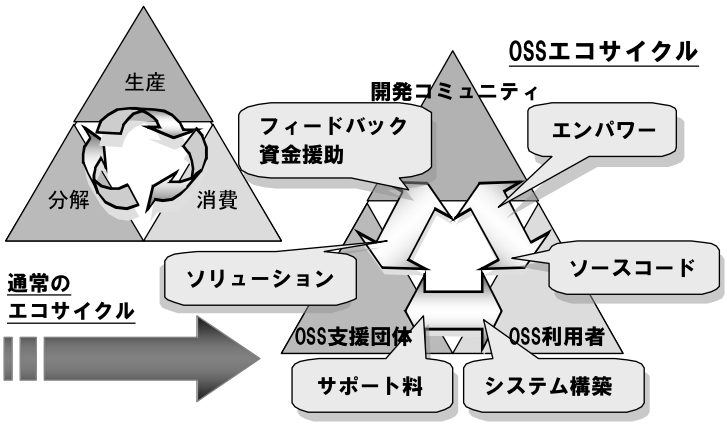

図 2　OSS エコサイクル

従来のプロプライエタリな製品の世界では，ソフトウェア開発者はサポート企業と同列の企業で雇われていることが多く，開発者と利用者の関係のみで，支援者が活躍する場所がなかった．そのため，開発者に市場が独占されることになり，生態系がうまく作られず，利用者は自由を失い，不便な仕様に我慢を強いられる結果となった．

一方，OSS の世界では，ソフトウェアがオープンソースであり，かつ，ライセンスや自由な精神によって守られているため，ソフトウェアの独占が起こりにくい．そのため，利用者と開発者の間の金銭による呪縛から解き放たれ，より自由な市場が形成されるようになる．その結果，両者をバックアップできる支援者の活躍するスペースができた．そして，支援者により，利用者は安心してソフトウェアを使えるようになり，開発者は資金援助を受けることができ，よい循環ができるようになった．これを OSS エコサイクルと呼ぶ．

支援者は，表 1 の説明の欄にもあるように，雑誌やブログなどのメディアに OSS 使用後の報告や感想を投稿したり，メーリングリストにバグ報告をしたりする人々も含む．この行為は，生産者へのフィードバックになる．もし，これがないなら，生産者が作るソフトウェアがよりよいものにならない．このとき，そのフィードバックがインターネット上に共有されることで，別の消費者にも利益をもたらす．これも一種のエコサイクルの要素である．

3.3　マニュアル作成

マニュアルなどのドキュメンテーションは，OSS の開発者にとってはモチベーションが低い．コード開発には，まるで世界を記述しているような万能感を持ったり，バグ追及には，パズルのピースをはめ込んでいくようなある種のトランス（高揚）感があったりする．完成した後の達成感だけではなく，その過程において，心躍るような感覚があると言われる．

一方，技術ドキュメンテーションでは，作成過程において，その一手一手が魅力に欠ける．OSS のドキュメントにひっそり入っているジョークは，モチベーションを自分自身で高めていることの表れとも言える．

それでも，ボランティアにより多くのドキュメントが作成されてきたが，Linux カーネルメッセージに関してはマニュアルが存在していなかった．

4. Linux カーネルのエラーメッセージ

本章では，Linux カーネルが出力するエラー・通知メッセージについて説明する．OSS メッセージペディアは，それらをドキュメント化の対象としている．

4.1 システムログ

まずは，Linux 利用者に，Linux カーネルからどのような仕組みでメッセージが届けられるかを見ていく．Linux カーネルでは printk という内部関数を使ってメッセージを出力する．以下は書式である．

```
printk(フォーマットストリング，可変引数);
```

ライブラリ関数 printf(3)と使い方は同じである．ただし，引数のフォーマットストリングに指定される文字列が，システムログ（以下，syslog）にそのまま出力されることになるので，syslog の形式に従う必要がある．そのため，フォーマットストリングの先頭には，ログレベル（プライオリティと呼ぶこともある）を指定する．例えば，ログレベルが「システムが使用不能」の場合は，“<0>”という文字列を先頭に付ける．実際の例を以下に示す．

```
printk(KERN_EMERG “Kernel panic - not syncing: %s¥n”, buf);
```

KERN_EMERG は ”<0>”という文字列のマクロである．実際には，「<0>Kernel panic - not syncing: Aiee, killing interrupt handler!」というようなメッセージがログに吐かれる．

次に，Linux のバージョン番号出力時の流れについて，図 3 にまとめる．

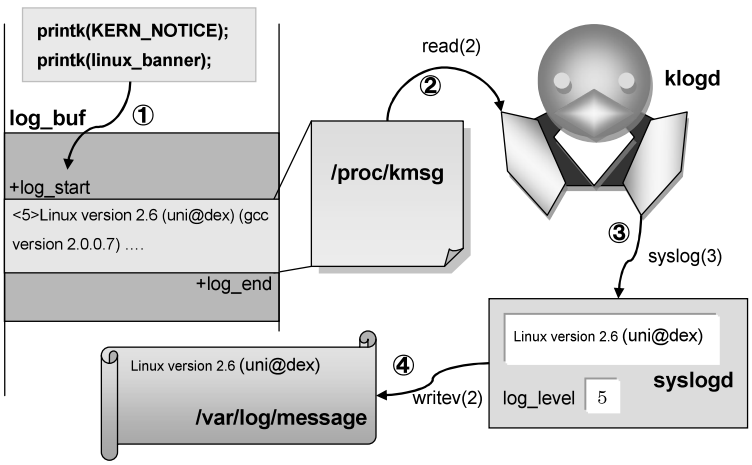

図 3　メッセージ出力の仕組み

カーネルは，カーネル内のログバッファ（log_buf）に書き込む（①）だけで，そのログは，klogd というプログラムが /proc/kmsg というファイルから取得する（オプションによってはシステムコールでメッセージを取得する）．このファイルは，proc ファイルシステム上のファイルであり，一般のファイルと異なり，ファイルの読み込み（②）時に，カーネル空間から klogd のバッファにコピーされる．klogd は，取得したテキストをライブラリ関数 syslog(3)で，syslogd プログラムにメッセージを送信する（③）．syslogd が，/var/log/message というファイルに書き込む（④）ことで，利用者から見ることができるようになる．

4.2 printk

Linux カーネル内では，単純に printk 関数を使う方法，ループを作ってメッセージを組み立てて出力する方法，あるいはそれぞれの開発者が独自にマクロを作る方法など，いくつかのバリエーションがある．ここでは，以下の 3 点を紹介する．

1) 単純な printk の使用

```
printk(KERN_INFO "CPU: Hyper-Threading is disabled¥n");
```

2) 複数行の printk で一つのメッセージを構成

```
printk (KERN_INFO "EXT3 FS on %s, ", sb->s_id);
if (EXT3_SB(sb)->s_journal->j_inode == NULL) {
        printk("external journal on %s¥n",
                bdevname(EXT3_SB(sb)->s_journal->j_dev, b));
} else {
        printk("internal journal¥n");
}
```

3) 独自のマクロ

```
#define ext3_debug(f, a...)                                          ¥
        do {                                                         ¥
                printk (KERN_DEBUG "EXT3-fs DEBUG (%s, %d): %s:",    ¥
                        __FILE__, __LINE__, __FUNCTION__);           ¥
                printk (KERN_DEBUG f, ## a);                         ¥
        } while (0)
```

4.3 問題点

Unisys 製メインフレーム OS である EXEC では，大部分のメッセージに番号を振り，メッセージ本体を記述したモジュールを別途用意しておき，メッセージ出力処理で，メッセージ番

号からメッセージ本体を参照する仕組みを持っている．つまり，それらのメッセージにはIDがついて管理されており，出力されたメッセージがどのメッセージであるかをすぐに判別できる．

しかし，Linuxは，メッセージをソースコードに埋め込むだけで，ソフトウェア全体としてメッセージを管理していない．また，UNIX® ライクなシステム上のプログラムが採用しているgettext[*13]による国際化も使っていない．そのため，メッセージの特定が困難である．また，特定できたとしても，ソースコードから意味を読解し，対処方法を調査するためのコストは大変高い．

このように，Linuxは，メッセージの出力方法自体にもマニュアル化の阻害要因を持っている．

5. OSSエコサイクルのハブとして

OSSエコサイクルの中でハブとして機能しているツールやサービスはいくつか存在する．例えば，バグ情報の障害データベースのBugzilla™ [*14]や，ソースコード共同開発環境を提供するsourceforge.netなどである．OSSメッセージペディアも，そのような存在を目指している．

そのためには，エコサイクルの登場人物たちに積極的に利用してもらう必要がある．本章では，それぞれの登場人物にスポットをあて，OSSメッセージペディアが提供する工夫点について説明する．

5.1 Linuxカーネル開発コミュニティ（生産者）

正確なメッセージマニュアル作りには，Linux本体からの支援が不可欠である．4.3節でも述べたように，外側からのメッセージの特定には，曖昧さと冗長さが入り込む．もちろん，一つずつ丁寧に，人手を介して，出力されるメッセージの内容を追っていけば，メッセージのカタログを作ることはできる．しかし，Linuxは常に成長しており，メッセージの内容は刻々と変わり，すぐに陳腐化してしまう．そのためには，プログラムによって自動的にメッセージカタログを作成する仕組みを備えていなければならない．

このような背景から，以下の観点でLinuxカーネル開発コミュニティ側に接触した．

A） OSSメッセージペディアの公開によりメッセージマニュアルの議論が起こること

B） Linuxがメッセージを特定するための仕組みを備えること

5.1.1 メッセージマニュアルの議論

まずは，OSSメッセージペディアの普及のために，ユニアデックスからLinux Foundation Japan[*15]の会議や，オープンソースカンファレンス，Linux Foundation Collaboration Summit（Kernel Messaging）[*16]などで紹介と問題提起のプレゼンテーションを行った．

Linux Foundation Japanの会議には，大変著名なカーネルメンテナー達が出席しており，メッセージマニュアルという地味な話題であったが，どのようにメッセージの特定を行うかなど，活発な議論がなされた．その結果，OSSメッセージペディアの存在が認知され，Linuxカーネルメーリングリスト内でメッセージマニュアルの必要性についての議論が始まった．さらに，アンドリュー・モートン[*17]氏から，Linux Foundationとしてこの問題に取り組みたいとの回答を引き出すことができた．

5. 1. 2　メッセージを特定するための ID

OSS メッセージペディアのプレゼンテーションで，メッセージに ID が付いていないと，メッセージの特定が困難になるという問題を折り込んだ．ここで，カーネル側が指し示すメッセージと，マニュアルのデータ（以下，コンテンツ）を提供する OSS メッセージペディアのような外部のサービスが指し示すメッセージが同一でなければならないという共通認識を得た．この認識なしに先に進むと，外部のサービスは陳腐化しやすくなり，サービスの存続が危うくなる．

上記の ID に関する認識と，メッセージマニュアルの必要性の認識が，Linux Foundation とその日本側の組織である Linux Foundation Japan との間で得られた結果により，linux-foundation.org で，lf_kernel_messages[*18] というメーリングリストが作成された．

メーリングリストでの議論の内容は，メーリングリストのアーカイブや，LWN の記事 “Getting the message from the kernel”[2] で詳細が見られる．現在までに，メッセージ ID 発番の仕組みの結論は出ていないが，これまでの議論の内容を表 2 で簡単に紹介する．

表 2　開発者との議論

タイトル	議論の内容
メッセージ ID	1) コード作成者が静的に割り振る 2) printk の引数のハッシュ値を ID とする 3) printk のフォーマットストリング[*19] 4) 既存の国際化 gettext の仕組みを利用する の 4 点があげられた
開発者に対して	開発の負荷になっては意味がないので，負担をかけない仕組みが求められる
コンテンツの格納場所	カーネルツリーには入れない
出力方法	a) UNIX ライクに，「ファイル名.行番号」を行頭に付加する b)「コンポート名.ID」を行頭に付加する
上記出力方法に対して	a) 行番号から ID に変換する仕組みが必要になる b) 端末に表示するまでに ID 値を計算する仕組みがいる
コンテンツの維持	なんらかの援助(資金など)が必要なのではないか

5. 2　OSS 利用者（消費者）

2007 年 4 月 25 日のニュースリリース後，2007 年 9 月末までに合計 16 万ページビューほどのアクセスがあった．また，はてな®ブックマーク[*20] や del.icio.us™[*21] などのソーシャルブックマークで約 700 件の登録があった．これは，想定した件数よりも多く，OSS に対して関心が高い反面，OSS の障害に対して不安を持っていることの示唆と捉えることもできそうである．ブックマークのコメントにも，長く続けて欲しいなどの意見があった．

また，スラッシュドット®[*22] にも取り上げられ，鋭い指摘を受けた．例えば，OSS を名称に含めているにも係わらず，Red Hat Linux™ のカーネル 2.6.9-34.EL しかサポートしていないという指摘や，まずは英語中心のサイトとして運用したほうが，より良くコンテンツを集められるというアドバイスなどである．

5.2.1　OSS メッセージペディアの工夫機能

次に，OSS 利用者との関係を育てていくために，OSS メッセージペディアが用意した機能を表 3 にまとめる．

表 3　OSS 利用者向けの機能

タイトル	内容
情報の共有	コンテンツをオンライン・データベースとして提供しているため，コンテンツの利用者や提供者の間で，情報のリアルタイム共有が可能である．
データの集中管理	情報が散在してしまうことを防ぐことができる．最新の情報を一箇所でウォッチすることができる．
コンテンツデータ	データは，Wiki を使ったシステムで採用されている簡易なマークアップをおこなっている．そのままでも十分構造化されているが，XTHML に変換し，そこに CSS[*23] を適用することで，Web ブラウザからマニュアル文書として閲覧可能にしている．
調査して欲しいメッセージ	調査して欲しいメッセージに対して，カウントアップできる機能を持っている．現在は，単純にカウントアップするだけの機能しか持っていないが，なぜ調査して欲しいのか，具体的にどの部分を解説して欲しいのかについて，コメントを入れられる機能の実装を考えている．

5.3　OSS 支援団体（分解者）

5.2 節と同様に，OSS 支援団体との関係向上のために，OSS メッセージペディアが持っている機能を表 4 にまとめる．

表 4　OSS 支援者向けの機能

タイトル	内容
コンテンツの編集	Wiki のように簡易マークアップ言語で記述する．また，編集データは履歴として保存され，誰がいつ，どのように編集したかを共有できる．PC プラットフォーム固有のアプリケーションを使う必要はなく，Web ブラウザがあれば編集できる．
オープンなライセンス	コンテンツや履歴データ全てを，現在は GFDL[*24] というライセンスをつけて公開している．このライセンスにより，データの再利用を可能にしている．
貢献度	貢献度にしたがい，コンテンツ提供者・団体名をタグクラウド[*25] 風に表示させるサービスを行っていたが，貢献度を単純には計れないことから，現在はサービスを停止している．
更新フィード	コンテンツの更新情報や調査して欲しいメッセージの更新情報を RSS や Atom フィードで提供している．コンテンツ提供者は，この調査して欲しいメッセージの情報をウォッチすることで，オンデマンドでコンテンツを提供できる．
コンテンツデータの一括提供	全コンテンツデータを TSV(タブ区切り)フォーマットのデータとして非同期に生成し，それをダウンロードできるサービスを提供している．
Web サービス	全コンテンツデータを Web サービス経由でアクセスする手段を開発した．その他にメッセージ検索のための Web サービスも計画している．これらのサービスは順次公開していく予定である．

6. 考察

本章では，OSS メッセージペディアが，OSS エコシステムの中でどのような役割を演じられるかについて再考する．次に，この OSS メッセージペディアの機能として，どのようなことが必要で，どのようなことを計画しているかをリストアップする．

6.1 OSS エコサイクルの中での役割

3.2 節で述べた OSS エコサイクル内での，提供・利用という推進力により，サイクルは快適に回るようになる．しかし，サイクル内の登場人物のネットワークをスムーズにつなげるためには，ハブとなるツールが必要である．OSS メッセージペディアは，その役割の一つとして機能するインフラを備えていると考える．

例えば，図 4 のように，サイクルのハブとして，OSS メッセージペディアを中心に置いてみる．ここでは，提供するものを角丸の吹き出しで，その結果として得られるものを四角の吹き出しで説明している．これを見ると，ハブを通して，お互いが良好な関係を築き，良好な OSS ライフを享受できることが分かる．もちろん，どこかに負荷が生じて，サイクルのバランスが崩れてしまう可能性はある．その場合は，何らかの調整や潤滑油の投入が必要だろう．

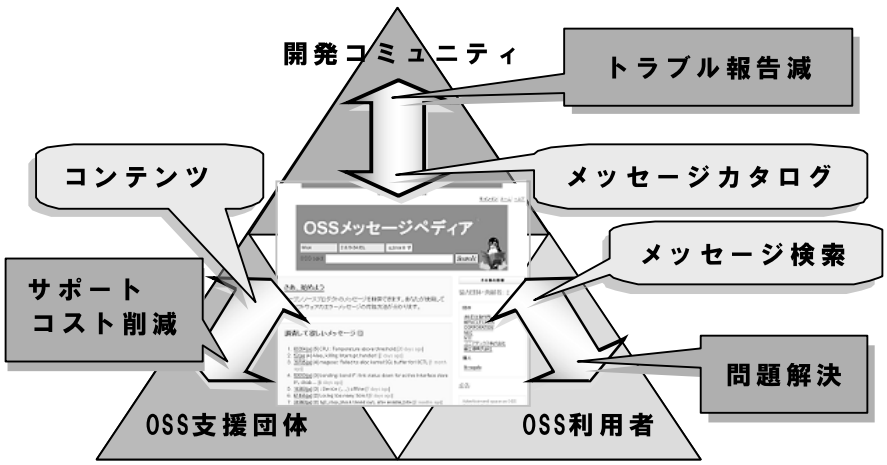

図 4　ハブとしての OSS メッセージペディア

OSS メッセージペディアがハブとして成功するためには，まず，ハブとつながる部分のプロトコルを制定する必要がある．5 章にて，その取り組みについて説明した．あとは，プロトコルの仕様書を作り，コードで実装していかなければならない．そのための作業項目を次節にて紹介する．

6.2 今後の予定

今後予定している作業の一部を以下に記載する．

- ID 割り振り方法を Linux カーネルコミュニティとともに制定する
- Linux 側のメッセージ ID 割り振りに対応させる
- 別の OSS（例えば，PostgreSQL）を追加する
- Web サイトの翻訳（特に，英語化）を行う

- コンテンツへのコメント，レーティングを可能にする
- OpenSearch，OpenID，Microformats の hReview に対応可能か調査する
- レコメンド機能（合わせて読んだほうがよいコンテンツのサジェストなど）を備える
- 検索精度を向上させる（レーティングの高いメッセージを上位にするなど）

上記のように，やるべきことは多い．また，OSS メッセージペディアとは別サービスとして，システムエラーやプログラムエラーの情報を集積するサイトを用意し，その情報から OSS メッセージペディアを事例検索に利用するというサービスも利用価値が高いだろう．これは Windows では既に提供されている機能である．

また，Bugzilla などの障害情報データベースとの連携も有用だろう．どのような操作でそのメッセージが出力されたのか，また，そのときの設定ファイルはどうだったのかなどの情報をコンテンツとすることで情報が充実していく．Bugzilla 側でも，対処方法を OSS メッセージペディアへのリンクで済ませることができるようになる．

OSS メッセージペディアの Web アプリケーションのソースコードは，sourceforge.net にて公開しており，コンテンツ編集者としての参加は，ossmpedia.org にて受け付けている．これにより，OSS メッセージペディアを通して，OSS のエコサイクルに支援者として関わることが可能である．

7. おわりに

新しいサービスは，破壊的イノベーティブな技術によって打ち出される．これは，最近のビジネスシーンでよく見られる考え方になっている．Linux が登場した当時，安価な PC で UNIX と同じソフトウェアを動かせることは，それまで安泰であったサーバビジネスの分野に風穴を開けた．

現在は，Linux や BSD™ 系の OS を搭載したサーバを利用している Google™ や Yahoo!® などの Web サイト，PostgreSQL を利用したデータウェアハウス・アプライアンスの Netezza™，OS のカーネルを独自カーネルから BSD 系のものに変更した Apple® などが，技術のイノベーションによって成功を収めている．

これらの企業の共通ポイントは，インフラの部分に OSS を採用している点である．そのインフラの上に，独自の技術を築き，他社との差別化を計っている．そこでの OSS は破壊的である必要はなく，安定であることが求められる．さらに，陳腐化を防ぐためには安定を優先した上で開発を継続していく必要がある．

これからは，上記のような企業だけではなく，一般の企業でも OSS をインフラとして利用するようになるだろう．そこでは必ず障害も発生する．そのときに，同じ問題を，別々のところで，同じように悩むことの無駄を省くことが OSS メッセージペディアや Bugzilla の利用価値となる．

OSS の開発や支援を通して，IT インフラにおける OSS の利用価値を高めることは企業価値を高めるだけではなく社会貢献にもつながる．今後も OSS メッセージペディアだけではなく，OSS エコサイクルの中で OSS の可能性を広げていきたい．

＊ 1　LDP：The Linux Document Project http://tldp.org/

＊ 2　Linux JF プロジェクト：Linux JF（Japanese FAQ）Project http://www.linux.or.jp/JF/
＊ 3　crackerjack：http://sourceforge.net/projects/crackerjack/
＊ 4　IPA：INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN の頭文字
＊ 5　Atom：http://www.ietf.org/rfc/rfc4287.txt，http://intertwingly.net/wiki/pie/FrontPage
＊ 6　Microformats：Microformats：XHTML の中の Machine Readable なマークアップ．http://microformats.org/
＊ 7　Hyper Estraier：http://hyperestraier.sourceforge.net/ 作成者のブログの内容が主に採用の理由
＊ 8　LXR：http://lxr.sourceforge.net/ 正規表現ではなく split 関数でコードをパースしている特徴ももつ
＊ 9　Catalyst：http://www.catalystframework.org/
＊10　VOX：他のサービスとの連携が充実したブログ・SNS サービス（Six Apart, Ltd）
＊11　ソースコード公開：https://ossmpedia.svn.sourceforge.net/svnroot/ossmpedia から取得できる
＊12　生態系：http://ja.wikipedia.org/w/index.php?title=%E7%94%9F%E6%85%8B%E7%B3%BB&oldid=14869884
＊13　gettext：http://www.gnu.org/software/gettext/gettext.html
＊14　Bugzilla：http://www.bugzilla.org/ Web ベースのバグ情報管理システム
＊15　Linux Foundation（旧 OSDL）Japan：http://www.linux-foundation.jp/
＊16　Collaboration Summit：http://www.linux-foundation.org/en/Kernel_messaging_agenda
＊17　アンドリュー・モートン：Linux カーネルのコアメンテナー
＊18　lf_kernel_messages：https://lists.linux-foundation.org/mailman/listinfo/lf_kernel_messages
＊19　フォーマットストリング：ソースコードのテキストイメージでフォーマット演算子も含む
＊20　はてなブックマーク：http://b.hatena.ne.jp/entry/4559637 ほか
＊21　del.icio.us：http://del.icio.us/url/1067b36d0282de1ab82e751a389fdfad ほか
＊22　スラッシュドット：http://slashdot.jp/linux/article.pl?sid=07/04/26/0428221
＊23　CSS: Cascading Style Sheets：HTML で構造化された文書のデザインを担当する
＊24　GFDL：GNU フリー文書利用許諾契約書 http://www.gnu.org/licenses/fdl.ja.html
＊25　タグクラウド：あるラベルのリストをデータの中の頻出度などによって視覚的な変化をつけること

参考文献　［1］ 日本ユニシス株式会社「HMP IX シリーズ・シリーズ 2200 EXEC 操作解説書 メッセージ編 レベル 46」2000 年
［2］ Jonathan Corbet「Getting the message from the kernel」http://lwn.net/Articles/238948/

執筆者紹介　前　原　志　好（Shiko Maehara）
1997 年日本ユニシス(株)入社後，米国 Unisys 製メインフレームのカーネルを担当する．2004 年 4 月ユニアデックス(株)に転籍．その後 OSS に着手．現在，ユニアデックス(株)サーバソフトウェア 1 部所属．