社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－SE－153 (1)
2006／7／10

アスペクト指向技術とSNMPによるコンポーネント稼動状態測定フレームワーク

楠 和泰† 玉井哲雄†

ソフトウェアの稼動状態を正しく測定することは，さまざまなメリットが考えられる．たとえば，システムの保守のために稼動状態を確認することや，ソフトウェアの使用状況を元に，ユーザから費用を徴収することなどがあげられる．特にユーザに対して，使用料金徴収の際に，使用した分だけ費用負担を求める「従量課金制」を実施する場合には，正確にソフトウェアの稼動状態を測定する必要がある．ソフトウェアの稼動状態を測定する問題については，多くの研究や提案がある．しかし，ほとんどが大規模エンタープライズ向けであったり，導入時に莫大な費用がかかるうえ，マルチベンダ未対応であったりする．そこで，当論文では，システムの規模を問わず稼動状態測定を実現させるフレームワークを提唱する．このフレームワークは容易に機能を組み込め，マルチベンダで稼動し，遠隔で監視することができ，ネットワークに負荷をかけずに管理できる利点がある．このフレームワークは，アスペクト指向技術を用いて，現行システムに対して大幅に変更を加えることなく，稼動状態測定機能を組み込むことを可能とする．また，ネットワークやネットワーク・デバイスの稼動状態測定で実績のあるプロトコルSNMPを採用することによって，ネットワークに負荷を与えず，単純・軽量に管理と監視をすることができる．さらに，このフレームワークを使用した稼動状態測定は，容易に従量課金を実行することが可能となる．システムのパーツを売買する市場である「流通コンポーネント市場」や，サーバに設置されたアプリケーションを提供するサービスである「ASPサービス」が注目される昨今，容易に従量課金を実行できることは，新たな商機を生み出すことに他ならず，市場をさらに活性化する役割を果たすものと期待される．

Framework for components' operational status evaluation using Aspect Oriented Programming and SNMP

KAZUHIRO KUSUNOKI† and TETSUO TAMAI†

Evaluating software operational status has a lot of merits. For example, it can be used to check if a system works correctly or to to charge fees based on the amount of usage. This is especially useful for evaluating software operational status, which is necessary in order to implement volume-limited charging. The Problem of evaluating software operational status has many proposals and solutions. Many of them, however, are limitatively applied only for enterprise system and the cost is very high. Moreover, a part of them can't be applied for the multi-vendor environment. So in this paper, we suggest a framework for evaluating software operational status regardless of system size. The profits this framework offers are firstly the easiness to build in to a system. Second it can work in a multi-vendor environment. Third it offers a measure to monitor components located in the remote network-transparently and its communication is light. This framework enables users to build the function of software operational status evaluation in to their system with few adding source code by using Aspect Oriented Programming. And it employees SNMP(Simple Network Management Protocol) which has actual performance on evaluating network operational status. In addition, evaluating software operational status by using this framework enables to implement volume-limited charging in easy way. Recently, COTS(Commercial Off-The-shelf Software) market and ASP(Application Service Provider) has been attracting more and more attention. In this situation, Being able to do volume-limited charging is expected to create new business chances and to make the market more active.

1. はじめに

昨今のIT業界のソフトウェア業界の動向としてコンポーネント指向の普及もあいまってCOTS(流通コンポーネント)市場，ASPサービスに注目が集まっ

† 東京大学大学院
The University Of Tokyo

てきている．COTS とは Commercial Off-The-shelf Software の略称であり，再利用可能でかつ他のアプリケーションにおいてもプラグアンドプレイが実現できるソフトウェア部品のことである．ASP サービスとは Application Service Provider の略称であり，あるサービス提供者が自身のネットワークのサーバ上にあるアプリケーションをネットワーク (通常はインターネット) を通して多数のユーザに利用させる形態のサービスを指す．

このようなサービスにおいて考えられるユースケースの一つに「ユーザへの課金をする」というものがある．この関心事を実現するにあたってまず考えられるのが (1) 定量課金 (2) 従量課金のどちらを採用するかという点である．システム開発という点から見た際に問題となるのは後者の場合であり，ユーザのサービス使用状態に基づいた課金をするためにはサービスの稼動状態の測定をすることが必要となり，当然「稼動状態測定をする」という新たなユースケースが発生する．

このユースケースを実現するためのソリューションや提案は数多くなされており，代表的な事例を挙げてゆくと IBM の Tivoli Omegamon®[1]，GRASP[2]，Glassbox Inspector[3] 等がある．Tivoli Omegamon® は大規模エンタープライズ向けソリューションであり，導入の際には高価でかつ大掛かりなものとなってしまう．更に当ソリューションを利用する場合，システムを構成するハードウェアやサーバが IBM 製のものに限られマルチベンダ型開発やオープンソース製品を利用したシステムには適用出来ない．GRASP は OGSI(Open Grid Services Infrastracture) のグリッド標準規格に則った会計サブシステムを実現している．GRASP は ASP サービスがグリッドを用いた規模の大きなサービスの場合に対して有用な研究であるが，グリッドを用いない (或いは用いることが出来ない) 程度の規模の ASP サービスでは利用できない．Glassbox Inspector は"AspectJ，JMX，Java Tiger の機能をフルに活かした"[3] 一種のプロファイリングツールである．このツールはオープンソースなので無料で利用出来る．このツールを使うことによってシステムモジュールレベルでの様々なデータをモニタリング出来る優れたツールであるが，このツールはモニタリングし収集されたデータをビューアーで確認することで稼動状態の健全性を確認するのが主目的のツールであるため，従量課金の実現をするためにこのツールを使うことは難しいと思われる．

以上に列挙したように，様々な先行事例があるものの，その多くが大規模エンタープライズに限るというような，スケーラビリティの問題が存在する．これにより従量課金を実現出来る企業はおのずとある程度以上の規模と資金を持ち得ている企業に限られる．そしてそこで当論文においては，スケーラブルであり，かつ容易に従量課金の仕組みを実現するためのフレームワーク "OSE フレームワーク"(Operation Status Evaluation フレームワーク) を提唱する．

2. OSE フレームワーク

最初に OSE フレームワークの目的と設計方針を説明した後，OSE フレームワークの構成を示し，各構成要素及びデータ通信の方法についての説明を加えた後，当フレームワークの応用例として従量課金への適用方法について述べる．

2.1 OSE フレームワークの目的

当論文で提唱する OSE フレームワークの目的はシステムの規模を問わず稼動状態測定機能を付与することである．当フレームワークを適用することにより，複数の VM により動作する，或いはネットワークを介した複数の端末からなるシステムからネットワークを全く用いない完全なスタンドアローンのものにいたるまで，システム全体またはコンポーネント単位における稼動状態を測定することが可能となる．そしてそのシステムの稼動状態に基づいた従量課金が実現できる．

2.2 OSE フレームワークの設計方針

OSE フレームワークは 2.1 項において述べた目的を果たすために，次の 3 つの要求を満たしている．

(1) 既存システムへの組み込みを容易に可能とする
(2) システムの規模を問わずに稼動状態対象のコンポーネントの計測を行う
(3) ネットワークの負荷を最小限に抑える

第 1 点目の要求を満たすためにまず第 1 に考えられるのは "いかに組み込み対象のシステムの変更を少なくするか" ということであろう．OSE フレームワークはこの要求を満たすために，アスペクト指向技術を活用した設計をしている．アスペクト指向技術を利用することにより，稼動状態測定のためのコードの追加を最小限に抑えることができるからである．具体的な説明は後述するが，コード中での稼動状態測定に利用するデータの通信を横断的関心事として，それらに対するアドバイスにおいて稼動状態測定処理を自動的にシステムに追加するという仕組みを用いて元システムのコード追加を最小限に抑えている．

第 2 点目及び 3 点目の要求への答えとして OSE フレームワークが採用した設計は，端末間及び VM 間において SNMP による通信を行う設計である．SNMP

を採用した理由としては3つの理由があり，1つ目は，SNMP は UDP による軽量通信であり，ネットワーク規模が大きくなっても比較的ネットワークにかかる負荷が低いことがよく知られている．

2つ目として，SNMP と稼動状態測定の親和性の高さである．実際にネットワーキングの世界においてSNMP によるネットワーク及びネットワーク機器の稼動状態測定がよく行われている．そこで当フレームワークではネットワークにおけるネットワーク機器を，システム中の稼動状態測定対象のコンポーネントと見立てることで SNMP をシステムの稼動状態測定に適用できるという考えのもとに設計を行っている．

3つ目は SNMP のデータ管理方式のシンプルな分かりやすさである．SNMP のデータ管理方式について説明すると，SNMP はデータを種類別にカテゴライズしたグループにまとめ，それらのグループをツリー形式で表現する方式である．このデータのグループは MIB(Management Information Base) という定義書により定義され，それぞれの MIB は RFC により標準化されている．次に SNMP のデータ管理通信を行う利点として必要あるいは有効なデータの種類の判別が容易になるという点である．後述する SNMP におけるデータ定義の MIB に則ってデータの管理が行われることによってユーザはどのようなデータが有効か，どのようなデータを使うことができるかを把握できる．☆

2.3 OSE フレームワークの特徴

OSE フレームワークの特徴として挙げられるのは次の3点がある．まず1点目は，アスペクトを用いたことにより元システムのコード追加を最小限に抑えている点である．

2点目にはユーザは稼動状態測定の対象としたいコンポーネントとデータを選択することが可能なことである．システム中のあらゆるコンポーネント及びデータを測定対象とするのではなく，目的とする課金に有用なコンポーネントとデータだけを取捨選択して稼動状態を測定できるのである．

3点目には OSE フレームワークは別の VM 上及び別のネットワーク上で動作している測定対象コンポーネントの稼動状態を全て透過的に測定可能であるため，スタンドアローンシステムのみならず分散システムや複数の VM の協調動作からなるシステムにおいても適用することが出来る．

☆ 有効でないデータとは MIB で定義されていないデータのことである．

2.4 OSE フレームワークの構成

図1は OSE フレームワークの構成の概要である．

図1の中において特に重要な責務を担っているのは

- SNMPSupport - 稼動状態測定の対象となるコンポーネントのインタフェイス
- OSEComponentManager - システム全体のコンポーネントを透過的に一括管理する
- OSEDataList - OSE フレームワークにおける MIB の管理を行う
- OSESNMPOperationAspect - SNMP 操作という横断的関心事へアドバイスを実行する
- OSEComponentAspect - SNMPSupport がインスタンス化された際コンポーネントを OSEComponentManager に登録する
- OSEComponentStatusStocker(CSS) - コンポーネントのデータを保持する
- OSEStatusAggregator(CSA) - OSEComponentStatusStocker のデータを定期的に収集する
- ComponentStatusEvaluator(CSE) - OSEComponentAggregator の集めたデータを利用し稼動状態を分析する

のクラスである．次節において上記のクラスがどのように動作し稼動状態測定を実現するかを詳細に見ることとする．

2.5 OSE フレームワークの仕組み

稼動状態測定を実現するために OSE フレームワークの各構成要素がどのように動作しているかのシーケンスを以下に示す．

(1) SNMPSupport を実現しているクラス，即ち測定対象となるコンポーネントをインスタンス化
(2) OSEComponentAspect が OSEComponentManager にコンポーネントを登録
(3) コンポーネントが MIB で定義されたデータを snmpset 命令を使って CSS に登録．登録しようとしたデータが MIB で定義されていないと OSEDataList が判断した場合はデータ登録を拒否
(4) あらかじめ定められた時間になると CSA が CSS に蓄えられたデータを収集
(5) CSE が CSA により収集されたデータを利用して稼動状態を分析する

上記の手順で示した通り，OSE フレームワークは稼働率測定対象コンポーネントの登録 - データの登録・

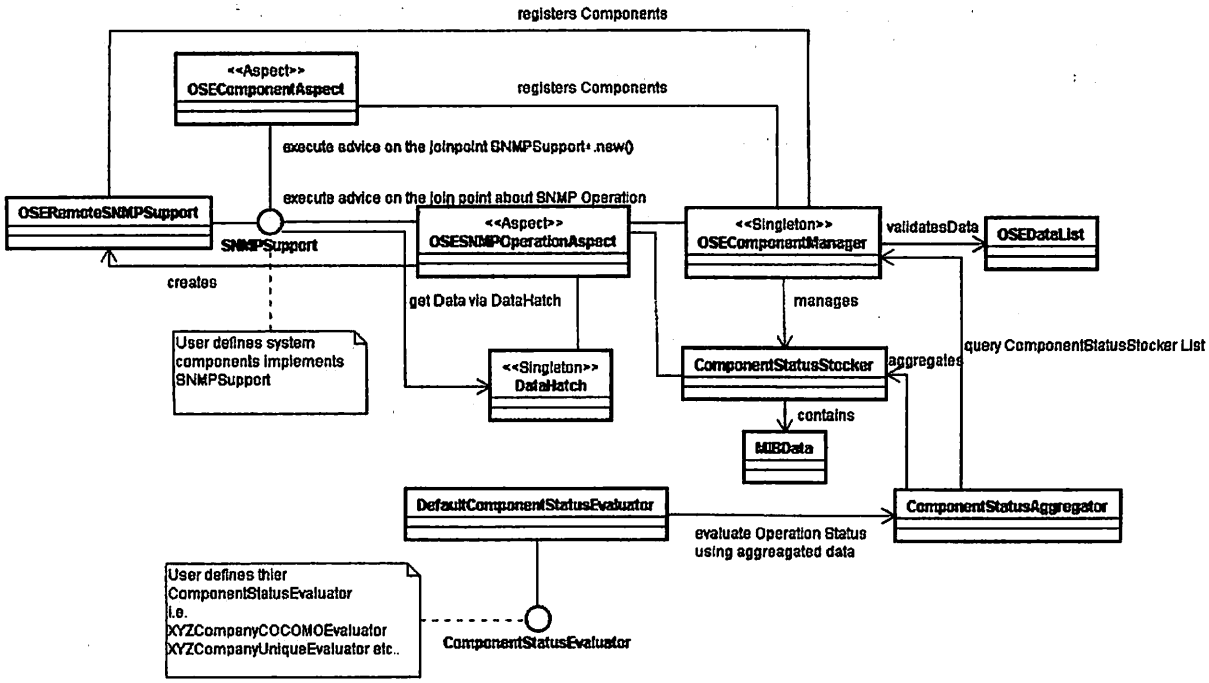

図 1　OSE フレームワーク概要クラス図

蓄積 - データの収集 - 分析 というフェーズを経て稼動状態測定を実現している．なお分析のフェーズの際，ユーザは会社，プロジェクト，システム毎に最適な稼動状態を測定するような CSE を実現したクラスを作成出来る．これにより柔軟に稼動状態の定義と分析をすることが実現可能となった．

2.6 OSE フレームワークにおけるアスペクト指向技術の適用

OSE フレームワークを既存のシステムに適用する際，アスペクト指向技術を適用することで既存のシステムのコード変更を最小限に抑えている．実際にアスペクト指向技術がどのように使われているかを見ることとする．図 1 にある通り，当フレームワーク中のアスペクトは 2 つあり，それぞれ以下のような関心事をまとめている．

- OSEComponentAspect - 稼動状態測定対象コンポーネントのインスタンスが生成される
- OSESNMPOperationAspect - 稼動状態測定対象コンポーネントが SNMP 通信を行う

上記 2 つのアスペクトがそれぞれの持つ関心事に対してアドバイスコードを実行することで稼働率測定対象コンポーネントの登録，データ登録・蓄積を実行する．ユーザが測定対象コンポーネントに SNMPSupport を実現させることと，測定対象としたいデータを snmpset() メソッドを呼び出すことで上記の操作は全て自動で行われる．

3. OSE フレームワークにおける SNMP 通信

OSE フレームワークにおいて，あるシステムにおいて稼動状態を測定する対象の部分をドメインということとする．そしてそのドメインは SNMPSupport インタフェースを実現したコンポーネントによって構成されている．(図 2)

ドメインはマスタドメインとスレーブドメインの 2 種類に分類され，マスタドメインは 1 つ，スレーブドメインは 0 以上存在する．このスレーブドメイン中のコンポーネントがマスタドメインへデータを登録，又はマスタドメインからデータを取得する際に SNMP による通信が発生する．言い換えるとマスタドメイン - スレーブドメイン間通信は実質上異なる VM 間で発生する通信であるともいえる．

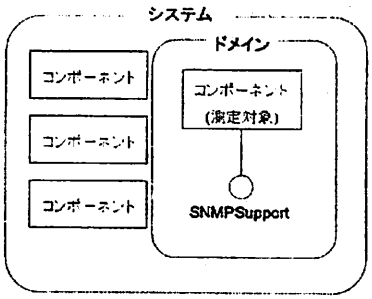

図 2 OSE フレームワークにおけるシステムとドメイン

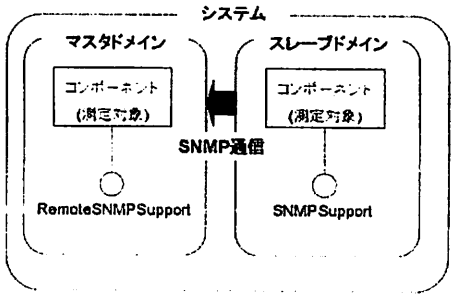

図 3 マスタドメインによる透過的コンポーネント管理

4. マスタドメインによる透過的コンポーネント管理

マスタドメインはスレーブドメインのコンポーネントをネットワーク透過的に管理することが可能である. スレーブドメインからマスタドメインへの SNMP 通信が初めて発生した場合, マスタドメイン内においてスレーブドメインのコンポーネントのリモートインタフェースが生成され (図 1 における RemoteSNMP-Support クラス), マスタドメインのコンポーネントマネージャ(OSEComponentManager) に登録される (図 3). これによりマスタドメインにおいて透過的なコンポーネント管理が可能となっている.

5. OSE フレームワークの適用による従量課金の実現

以上, OSE フレームワーク自身の仕組みについて述べてきたが, 当セクションにおいては OSE フレームワークを実際どのようにして従量課金の実現のために適用するのかについて述べる. まず実際に適用するにあたってユーザが行うことは以下の通りである.

(1) そのシステムにおける稼動状態の定義をする
(2) システム中におけるドメインを定める
(3) 定義された稼動状態を示すために必要なデータを選ぶ
(4) 稼動状態定義に従った従量課金のビジネスロジックを CSE に実装する
(5) OSE フレームワークを組み込んだシステムを稼動させる

以上 5 ステップが稼動状態測定に基づく従量課金実現のステップとなる. この具体例を次項のケースワークを通して説明する.

6. ケースワーク : ウェブ上のファイルストレージサービス

ウェブ上のファイルストレージサービスにおける稼動状態測定に基づく従量課金実現までを第 5 項で示した手順に沿って見てみると,

(1) このサービスにおける稼動状態の定義を "システム利用者のアップロード/ダウンロードしたバイト数" とする.
(2) ドメインはアップローダ, ダウンローダのロジックを担うコンポーネントであると定める.
(3) 必要なデータはユーザの ID となるものと受送信バイト数であると定める. このケースワークの場合ユーザの ID はサービス利用時に発行したユーザ ID と仮定する.
(4) ユーザ ID 毎に 1ヶ月分のファイル送受信バイト数から請求金額を算出. そしてその帳票を作成するロジックを実装する CSE を作成する.
(5) CSA は 24 時間に一度,CSE は一ヶ月に一度起動する設定でシステムを稼動させる.

上記のステップを経てシステムが一月の稼動状態定義に基づいた従量課金をユーザ毎に行い, そして請求帳票を作成する仕組みが出来上がる.

7. 課題と今後の拡張

現在の OSE フレームワークではマスタドメインとスレーブドメインの通信において SNMP での通信を採用しているのは先にも述べたとおりであるが, そのために SNMP の持つ欠点を OSE フレームワークが引き継いでいる. SNMP の持つ欠点は 2 点あり, 1 点目はセキュリティの脆弱性, 2 点目は集中管理型データ管理の弊害, 即ちデータを収集する責務を持つ端末の負荷が過剰になる恐れがあるという点である.

この 2 点について考察してみると, 第一点目の問題は SNMP リクエスト, SNMP レスポンスの全てのパケットは暗号化されないまま送出される. ASP といったローカルネットワーク内のみのドメインで構成さ

れるサービスならば問題はないと思われるが，WEB サービスや COTS のようにに WAN を介して SNMP パケットが送受信するような場合はセキュリティ上の問題が発生する．この問題への対策として，暗号化した SNMP の規格が提唱されている[5]．OSE フレームワークにおいても SNMPv3 の規格に則り，暗号化 SNMP をサポートする必要がある．

第二点目は SNMP は元々複数台による分散データ管理をサポートしておらず，単一の端末において全ての SNMP エージェントから情報を収集・管理をすることを前提としている．このような問題に対しては ILM[6]，AgentX[7] といった分散情報収集管理の手法が提案されている．OSE フレームワークにおいても現在の設計ではマスタは稼動状態管理ドメイン一つにつきマスタは一つであるという前提条件がある．これを稼動状態管理ドメインが複数のマスタを持ち，複数マスタ間による透過的な情報管理を行えるように変更することにより，分散情報収集管理が可能となる．

謝辞 研究について様々な助言をしていただいた増原英彦先生を始め，玉井研究室及び増原研究室の皆様に対し，謹んで感謝の意を表します．

参 考 文 献

1) IBM Tivoli OMEGAMON®.
http://www-306.ibm.com/software/tivoli/products/omegamon-zvm/
2) Grasp, The Grid-based Application Service Provision (GRASP) project
http://eu-grasp.net/english/default.htm
3) glassbox-inspector, Glassbox Inspector Project
https://glassbox-inspector.dev.java.net/
4) RFC 1157 A Simple Network Management Protocol (SNMP). IETF.org
http://www.ietf.org/rfc/rfc1157.txt
5) RFC 2570 SNMPv3. IETF.org
http://rfc.net/rfc2570.html
6) Rajesh Subramanyan, Josè Miguel-Alonso, Josè A. B. Fortes,
"A scalable SNMP-based distibuted monitoring system for heterogeneous network computing"
ACM/IEEE conference on Supercomputing (2000)
7) M. Daniele, B. Wijnen, D. Francisco
"Agent Extensibility(AgentX) Protocol", RFC2257 (1998)
http://www.isi.cdu/in-notcs/rfc2257.txt
8) P. Miller : CP/IP Explained, Digital Press (1997).
9) Steve Maxwell : SNMP ネットワーク管理ツール，翔泳社 (2001).
10) 斗光佳輝 : SNMP ツール開発テクニック，CQ 出版 (2003).
11) 玉井 哲雄 : ソフトウェア工学の基礎，岩波書店 (2004).
12) D. Pilone : UML Desktop Referrence, O'Reilly (2003).
13) J. D. Gradecki, N. Lesiecki : Mastering AspectJ, Wiley Publishing (2003).
14) R. Laddad : AspectJ In Action, Manning (2003).
15) G. Kiczales, J. Lamping, A. Mendhekar, C. V. Lopes, J. Loingiter, J. Irwin.
"Aspect-Oriented Programming"
European Conference on Object-Oriented Programming(ECOOP) (1997)
16) Component Source
http://www.componentsource.co.jp/index.html
17) Component Square
http://www.c-sq.com/