情報処理学会論文誌 Vol.55 No.2 659–669 (Feb. 2014)

XMLルールのコンパクト化に基づく監視コンポーネント向け軽量ルールエンジン

阿倍 博信[1,a)] 中島 宏一[1] 峯村 治実[1]

受付日 2013年4月24日, 採録日 2013年10月9日

概要：本論文では，監視カメラなどの監視コンポーネントの内部で動作ルールに基づき自律的にイベントを処理することによりレスポンス向上を目的とした軽量ルールエンジンについて述べる．ルールエンジンの開発において，組み込み CPU での軽量動作を考慮して，システム設定時に XML 形式で記述したルールを独自形式にコンパクト化する方式を採用した．組み込み Linux 環境上に軽量ルールエンジンを実装した評価システムを開発し，イベント処理性能とメモリ使用量について評価実験を行った結果，その有効性について確認することができた．

キーワード：イベント処理，監視コンポーネント，ECA，ルールエンジン，XML，コンパクト化

A Light-weight Rule Engine for Video Security Device by Compacting XML Rule

Hironobu Abe[1,a)] Koichi Nakashima[1] Harumi Minemura[1]

Received: April 24, 2013, Accepted: October 9, 2013

Abstract: In this paper, we describe a light-weight rule engine that are intended to improve the response by handling the event in accordance with the rules operate autonomously within the video security device, such as a video security camera. In the development of the rule engine, which adopted the compact in a proprietary format rules that consider the behavior of light in the embedded CPU, written in XML format during system configuration. The results to develop a prototype system that implements the light-weight rule engine on Linux embedded environment, the evaluation experiment about event processing performance and memory usage, we were able to confirm its effectiveness.

Keywords: event processing, video security device, ECA, rule engine, XML, compacting

1. はじめに

防犯意識の高まりにより，監視カメラを用いた映像監視システムが普及しつつある．特に，映像符号化技術，IP ネットワーク技術の進歩により，高精細映像のデジタル IP 伝送を可能としたネットワーク映像監視システムが注目されている [1]．映像監視業務では，監視の見落としなどによる重大事故を発生させないための機能強化が求められる一方，誤報の抑止による業務効率化，コスト削減などのニーズが高くなっている．

上記要求に対するアプローチとして，近年のセンサ技術や映像処理技術の発展により，映像監視システムにおいても，各種センサからの出力や監視カメラで撮影した映像データの映像処理結果などをもとに様々なイベントを発生させ，その結果を処理することによってシステムの効率化が可能となりつつある．たとえば，映像処理結果と関連するセンサの出力を組み合わせることによりイベントの信頼性の向上を図ることができる [2]．

一般的な映像監視システムでは，各種センサのほか，組み込みシステムを前提とするカメラ，レコーダ，ビューワなどの装置（以下，監視コンポーネント）と制御 PC から

[1] 三菱電機株式会社
Mitsubishi Electric Corporation, Kamakura, Kanagawa 247–8501, Japan
[a)] Abe.Hironobu@cs.MitsubishiElectric.co.jp

構成される．制御 PC で監視カメラや各種センサで発生したイベントを処理し，その結果をもとに，各コンポーネントの制御（エンコード制御，蓄積制御，表示制御など）を行う．この場合，監視カメラやセンサなどから発生したすべてのイベントを制御 PC で処理することになるため，システムコストが高くなるとともにイベントが集中したときのシステムのレスポンス低下が課題となっていた．

具体的には，カメラ台数が 50〜100 台程度の中大規模システムを想定した場合，たとえば 100 台のカメラに対するイベント処理を制御 PC1 台で同時に実行すると，イベントの輻輳によるレスポンス低下の発生確率が高くなってしまう．

上記課題に対応し，カメラやレコーダなどの監視コンポーネントに搭載し，制御 PC を使用せずにコンポーネント内部でイベント処理を実現することを目的とした軽量ルールエンジンを開発した．

今回開発した軽量ルールエンジンを適用した映像監視システムでは，システム設定時に，設定用 PC で XML 形式で記述されたルールを XML パーサで解析して得られた DOM（Document Object Model）形式のデータを，動作ルールに特化した独自のコンパクト形式に変換して監視コンポーネントに格納しておく．システム運用時には，CPU やメモリなどのリソースの限られた監視コンポーネントにおいて，コンパクト形式のルールを直接扱うことにより，イベント処理におけるメモリ使用量の低減化とイベント処理の高速化の両立を実現する．

本論文では，2 章で関連研究，3 章で要求条件，4 章で XML ルールのコンパクト化方式，5 章で評価システムの開発，6 章で評価，7 章で考察について述べ，最後にまとめを行う．

2. 関連研究

一般的にルールエンジンはサーバ PC 上で動作するビジネスルールエンジンのことを指し，代表的なオープンソースの実装としては JBoss Drools [3] や OpenRules [4] が存在する．これらは CPU やメモリなどのリソースの十分なサーバ PC 上での動作を前提としており，リソースの限られた監視コンポーネント上での動作は考慮されていない．

寺田らはウェアラブル PC 上で動作するルールエンジンである Wearable Toolkit [5] を開発した．こちらも組み込み機器と比較してリソースの十分な Windows 環境上での動作を前提としているため，リソースの限られた監視コンポーネント上での動作は考慮されていない．

本論文では，ルールの記述は汎用的な XML 形式を前提とし，リソースの限られた監視コンポーネント上での動作を目的としており，その観点から，ルールに適した XML データの圧縮方式がポイントとなる．

一般的に XML データを圧縮する場合，汎用なテキスト圧縮ツールである gzip [6] や XML データ向けの XML 圧縮ツールである XMill [7] が使用される．これらの方式は XML データのアーカイブまたはネットワーク伝送時の帯域幅削減が目的であり，圧縮された状態での問合せ処理ができないため，本論文の目的には適さない．

また，最近では，圧縮された XML データに対して，XML データ全体を圧縮解除せずに，必要な部分のみを解凍することにより，直接問合せを行う方式の研究が行われている．XML データの圧縮形式としては，圧縮後のデータ全体の構造がオリジナルの XML データの木構造に直接対応するものを同型圧縮，対応しないものを非同型圧縮と呼ぶ [8]．

本論文では，リソースの限られた監視コンポーネント上での動作を考慮し，データ圧縮後のデータ全体の構造がオリジナルの XML データの木構造に直接対応する同型圧縮で，問合せ時に XML データの圧縮解除が不要な方式に着目する．これらを目的とした技術の例としては，XGrind [9] や XPress [10] が存在する．

XGrind は，基本的に，元の XML データの木構造を残したまま，各要素中のテキストデータを個別に圧縮する．データの木構造が圧縮データ中にほぼそのまま残っているため，圧縮したデータを先頭から解凍しながら問合せをする処理に向いている．XGrind は必要な部分のみ展開して処理を行う SAX パーサには適しているが，ルール処理を行う場合は，XML データ全体を DOM 形式としてメモリ内に展開する必要があるため，本論文の目的には適さない．

また，XPress も直接問合せが可能な XML 圧縮形式で，ルートから各要素へのパスの情報を算術符号を用いて圧縮する．算術符号では，ある文字列 S_1 がある文字列 S_2 の接頭辞になっている場合，そのことが S_1 と S_2 を圧縮したもののみを見て判定できる特長があり，効率の良い問合せ処理を実現している．しかし，XPress も XGrind と同じ理由から本論文の目的には適さない．

一方，同型圧縮の標準化の流れとして，W3C において，XML をバイナリ化して圧縮し，転送速度や必要なメモリ容量を小さくした標準フォーマットとして，EXI [11], [12]（Efficient XML Interchange）が勧告された．EXI は，XML の情報をバイナリ化してコンパクトな表現に置き換え，さらに圧縮をかけることにより，XML 文書の文書サイズを小さくすることが可能となる．しかし，EXI ストリームの形式は，基本的に SAX や StAX で利用されているイベントを直列化した形式をベースとしているため，イベント駆動のアプリケーションでの利用には適しているが，本論文のルールエンジンで必要である DOM 形式のアプリケーションには適さない．

特定の目的を持つ XML データの圧縮方式として，吉田らは，CSV 形式のデータ圧縮を目的とした XML CSV 圧縮法 [13] を開発した．この方式は，レコード形式の大容量 XML 文書に対し，アクセス面から冗長な複数の要素を

CSV 形式でまとめることにより，データ処理の効率化を実現したものである．XML CSV 圧縮法は，レコード形式の大容量 XML データに対する効果は認められているが，IF 文が中心となるルールイベント処理には適さない．

また，組み込み機器上での XML データの処理性能改善に関して，小林らは携帯電話上で主にコンテンツを中心とする XML 処理を高速化する技術である XEUS [14] を開発した．XEUS では，任意の XML データを対象とし，そのスキーマ情報を含む符号化テーブルをあらかじめ別の XML データとして定義し，符号化時にパース処理と符号圧縮処理を行うことで，リソースの限られた携帯電話での高速復号を実現している．

今回，携帯電話よりさらにリソースの限られた監視コンポーネントに XML ルール処理を実装する必要があり，また，コンテンツ処理も必要ないことを考慮し，XML パーサで解析した後のデータを，動作ルールに特化した独自のコンパクト形式に変換し，監視コンポーネント上で直接処理することにより，省リソース環境での動作を図る．

3. 要求条件

今回，軽量ルールエンジンの実装対象として，監視コンポーネントの中で最もリソースの限られているネットワークカメラを選定した．以下，ネットワークカメラへの実装を前提に，軽量ルールエンジンの要求条件を検討する．

具体的には，映像監視システムにおけるイベント処理を想定して以下のシナリオを設定し，要求条件の検討を進めていくこととした．

- **シナリオ 1**
  一般的な映像監視システムにおけるアラーム監視を想定し，特定のアラームイベント発生を受け，カメラのフレームレートを変更する．一般的な使用法は，通常時はフレームレートを低く設定しておき，アラーム発生時にはフレームレートを高く設定する．
- **シナリオ 2**
  鉄道などの移動体に搭載したネットワークカメラの映像データの地上システムへのモバイル回線経由での伝送を想定し，定期的に無線装置などから帯域情報を取得し，ルールに基づき，カメラの符号化パラメータを変更する．

関連研究である文献 [5] では，開発したルールエンジンを用いてルール数 127 個の複雑な大規模システムを構築した事例が紹介されている．文献 [5] 中ではシステムのスケーラビリティについては言及されていないが，イベント処理に対するリアルタイム要求がなければ，リソースの許す限り，ルールの追加による複数シナリオの並列動作が可能であると考える．

上記を考慮し，今回開発する軽量ルールエンジンにおいても，動作するシナリオで設定したルールどうしが競合していない限り，複数シナリオの並列動作が可能な形での設計を前提とする．ただ，リアルタイム要求の高い監視コンポーネントに実装する場合は，後述する性能要件を考慮しておく必要がある．

(1) **性能要件**
一般的にネットワークカメラの映像データのフレームレートは最大 30 fps（frame per socond）である．監視コンポーネント上のルールエンジンで動作ルールを解釈し，外部や内部で発生したイベントに対する制御をリアルタイムに行う必要がある．このため，イベントの発生から制御開始までの目標応答性能を，33 msec（1 フレーム）以内に設定した．

(2) **メモリ要件**
組み込み機器であるネットワークカメラへの実装を前提として，動作ルールを事前にコンパクト化することにより，通常の XML パーサを使用する場合と比較してメモリ使用量の低減が期待できる．オーバヘッドも考慮し，目標メモリサイズを一般的な XML パーサを使用した場合の 20%削減に設定した．

メモリ要件の設定理由について下記のとおり説明する．

一般的に DOM 形式に対応した XML パーサは起動するだけで一定のメモリ使用量を必要とする．また，XML データのサイズが大きくなればなるほど，そのメモリ使用量は大きくなる．たとえば，本論文とは目的が異なるが，中島らの開発した Web アプリケーション向けの XML 処理高速化技術である SplitDOM の開発において，従来の DOM と SplitDOM のメモリ使用量の比較を行っている [15]．その結果，XML パーサを起動しただけでも 200 KB 程度のメモリ使用量が必要となっている．今回，テキスト形式の XML データをバイナリ化することにより一定の圧縮効果は期待できるものの，DOM 形式に対応するためにはある程度のメモリを確保する必要があると考え，上記のとおり目標メモリサイズを設定した．

4. XML ルールのコンパクト化方式

3 章で設定した要求条件をふまえて，XML ルールのコンパクト化方式について検討した．

4.1 ルールファイル

本ルールエンジンでは，動作ルールは ECA（Event Condition Action）ルールとして定義し，XML 形式で記述する．本論文では XML 形式で記述された ECA ルールを XML ルールと定義する．ECA ルールとは，アクティブデータベースにおいて自動的に実行する処理を定義するためのルール定義であり，ルールの実行のトリガとなる Event，ルールが実行された際に確認される条件 Condition，条件を満たした際に実行する処理 Action から構成される [16], [17]．以下，ECA ルールの定義，XML ルールのタグ構成，XML

表 1 Action の例
Table 1 Example of action.

対象デバイス	カテゴリ	Action の例
カメラ	カメラ制御	パン・チルト・ズーム，フォーカス，絞り
カメラ	エンコード制御	符号化方式，フレームレート，圧縮率，解像度
レコーダ	蓄積制御	蓄積開始・中断・再開・終了
ビューア	表示制御	表示ソースの選択，表示開始・終了，ポーズ

ルールの例，XML Schema によるルール定義について説明する．

4.1.1 ECA ルールの定義

4.1.1.1 Event，Condition の記述

監視コンポーネントでのルール実行のトリガとなる Event は，様々なセンサ（温度センサ，人感センサ，接点，タイマーなど）からの入力値や，システムの状態（監視コンポーネントのリソースなど）をトリガとして発生する．Event は監視コンポーネントの種類別に定義する必要があるため，対象とするデバイスに依存する．たとえば，3 章で設定した要求条件では，シナリオ 1 では，「特定のアラームイベント発生」が，シナリオ 2 では，「無線装置からの帯域情報の取得」が，Event に相当する．

上記 Event をトリガとして，その Event に対応したルールが実行される．その際に確認される条件を Condition として定義する．Condition は，基本的に Event ターゲット（target），関係演算子（operator），値（value）の 3 つの情報で定義する．すなわち，target の値が value の値と operator で示した関係を満たしたときに，条件を満たしたこととなる．たとえば，3 章で設定した要求条件では，Event も含めた形で記述し，シナリオ 1 では，「特定のアラームイベント発生を受信」が，シナリオ 2 では，「無線装置から取得した帯域情報の数値と設定条件を比較」が，Condition に相当する．

Condition は，上記のような単純な条件に加えて，not，and，or などの複雑な条件も記述することができる．また，then を導入することにより，複数の条件が発生する時間的な関係を記述することができる．then では，複数の条件が発生する時間を定義する必要があり，時間の値（time），時間単位（timeUnit），関係演算子（operator）を定義する．

4.1.1.2 Action の記述

監視コンポーネントにおいて，条件を満たした際に実行する処理である Action は，対象デバイスごとに異なるため，それぞれ定義する必要がある．**表 1** に Action の例について示す．

Action として，デバイスターゲット（target），パラメータ（parameter）を定義する．パラメータは複数設定することも可能である．また，複数のルールから同じパラメータを使う可能性もあるため，対象となるパラメータをパラメータセット（parameterSet）として定義してパラメータセット ID（parameterSet ID）を設定し，その parameterSet ID を参照することで利用できるようにする．

4.1.2 XML ルールのタグ構成

4.1.1 項で定義した Event，Condition，Action を記述するために，XML ルールのタグ構成を下記のとおり定義する．

4.1.2.1 ルール全体のタグ記述

ルールは，ルールの集合である ruleSet として表される．

ruleSet タグ 1 つ以上の rule と，ruleSet 内部で共通的に使用する 0 個以上の parameterSet を定義する．

rule タグ 1 つの条件式を示す expression と，1 つ以上の action を定義する．

4.1.2.2 Event，Condition 関連のタグ記述

expression は，単一条件である condition と複合条件である not，and，or，then のいずれかで定義される．また，expressions は複数の expression を示す．

condition <condition operator="operator" target="target" value="value" /> の形で表される．operator は，EQ(=)，NE(≠)，LT(<)，LE(≤)，GT(>)，GE(≥) である．

not <not> expression </not> の形で表される．

and <and> expressions </and> の形で表される．

or <or> expressions </or> の形で表される．

then <then operator="operator" time="time" timeUnit="timeUnit" > expressions </then> の形で表される．operator は EQ，LT，LE，GT，GE，time は非負の整数，timeUnit は day，hour，min，sec，msec である．

4.1.2.3 Action 関連のタグ記述

action <action target = "target" > parameter or parameterRef </action> の形で表される．

parameter <parameter> parameter </parameter> の形で表される．

parameterRef <parameterRef id="parameterSet ID" /> の形で表される．

parameterSet <parameterSet id="parameterSet ID" > parameter </parameterSet> の形で表される．parameterSet ID は，ruleSet 内で一意であることを保証する必要がある．

4.1.3 XML ルールの例

XML 形式で定義したルールの例を，**図 1** に示す．図 1 において，最初に定義されたルールがシナリオ 1 の例，次に定義されたルールがシナリオ 2 の例となる．

4.1.4 XML Schema によるルール定義

XML では，スキーマを定義しないで自由にデータを記述することもできるが，今回の目的では，ルール定義をすべての監視コンポーネントにおいて同一の書式で行

```
<?xml version="1.0" encoding="shift_jis"?>
<ruleSet>
  <rule>
    <condition operator="EQ" target="sensor01"
    value="ON" />
    <action target="camera01">
      <parameterRef id="parameter01" />
    </action>
  </rule>
  <rule>
    <and>
      <condition operator="GT" target="wimaxband"
      value="330000"/>
      <condition operator="LE" target="wimaxband"
      value="670000"/>
    </and>
    <action target="camera01">
      <parameter>framerate=3, fcrate=20</parameter>
    </action>
  </rule>
  <parameterSet id="parameter01">framerate=30
  </parameterSet>
</ruleSet>
```

図 1 XML ルールの例
Fig. 1 Example of XML rule.

うため，厳密なスキーマを定義する必要がある．XML のスキーマ定義は，データ型や名前空間の面で優れている XML Schema [18] を用いる．XML Schema を用いることで，ルールファイルを DOM 形式で展開する際に，あわせて文法チェックすることが可能となる．

4.2 コンパクト形式への変換

4.2.1 概要

DOM 形式で XML データを保持すると，XML データ内のすべての要素・属性・テキストデータにランダムアクセスできるメリットはあるが，使用するメモリ量が多くなるという欠点があるため，リソースの限られている監視コンポーネントには適していない．また，ECA ルールとは無関係な情報が含まれているため，冗長である．このため，ECA ルールに特化したコンパクトな独自形式への変換を行う．

まず，ルール内の各要素の個数や文字列データの長さをカウントして，コンパクト形式に必要なメモリサイズを計算する．以下の要素についてカウントを行う．

rule 数 <rule> 要素の個数
condition 数 <condition> 要素の個数
action 数 <action> 要素の個数
parameterSet 数 <parameterSet> 要素の個数

表 2 コンパクトファイル用タグコード
Table 2 Tag code for compact file.

タグ名	値	タグ名	値	タグ名	値
condition	20	EQ	40	day	50
not	21	NE	41	hour	51
and	22	LT	42	min	52
or	23	LE	43	sec	53
then	24	GT	44	msec	54
parameter	30	GE	45		
parameterSet	31				

文字列長 テキストノード（parameter）の文字列長
属性値サイズ “target”，“value”，“time” の文字列長
論理式要素数 <not>，<and>，<or>，<then> 各要素の個数

次に，メモリサイズの算出結果に基づき，メモリを確保し，DOM データの木構造を順次たどりながら，次に示すコンパクト形式への展開を行う．

4.2.2 コンパクトファイルの形式

4.2.2.1 概要

コンパクトファイルの概要について以下に示す．

- メモリ内に展開された DOM データの木構造を同型のままでバイナリ化した形式とする．
- 組み込み機器の CPU における例外発生を防ぐため，2 バイトの値は 2 バイトアラインメント，4 バイトの値は 4 バイトアラインメントとする．
- 各ポインタは，ファイル先頭からのオフセットをバイト数で表す．
- condition No.，action No. などの番号は，0 から始まる整数とする．
- parameterSet ID は，parameterSet No. として文字列から番号に変換する．
- operator などは 1 バイトのタグコードで表す．**表 2** に各タグの値を 16 進数で示す．
- target や parameter などの可変長文字列データは \0 で終端する．

4.2.2.2 全体構成

コンパクトファイルの全体構成について**図 2** に示す．

図 2 に示したとおり，コンパクトファイルはコンパクトファイルのバージョン，ルール（rule），条件式（condition），アクション（action），パラメータセット（parameterSet）の 4 つのテーブルとそのポインタにより構成される．

4.2.2.3 rule テーブルの内容

- rule 数 NR
- rule へのポインタ × NR
- rule × NR

rule

- expression
- action 数 NA

- action No. × NA

expression

以下のいずれかとする．なお，矩形で囲まれたデータはタグコードとして記録される．

condition condition，condition No.

not not，expression

and and，expression 数 NE，expression へのポインタ × NE，expression × NE

or or，expression 数 NE，expression へのポインタ × NE，expression × NE

then then，operator，time\0，timeUnit，expression 数 NE，expression へのポインタ × NE，expression × NE

operator

以下のいずれかとする．

EQ，NE，LT，LE，GT，GE

timeUnit

以下のいずれかとする．

day，hour，min，sec，msec

4.2.2.4 condition テーブルの内容

- condition 数 NC
- condition へのポインタ × NC
- condition × NC

condition

operator，target \0，value \0

4.2.2.5 action テーブルの内容

- action 数 NA
- action へのポインタ × NA
- action × NA

action

- target \0
- parameter 数 NP
- parameter へのポインタ × NP

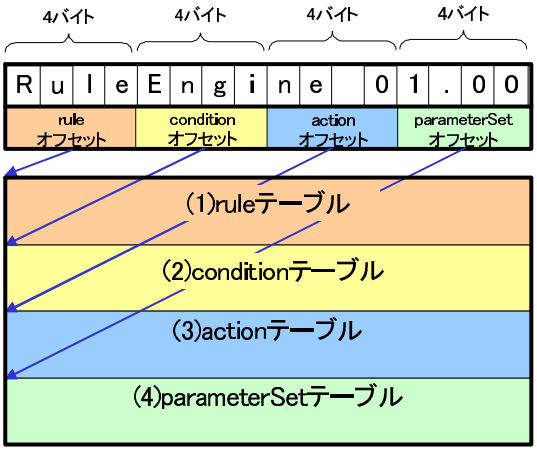

図 2 コンパクトファイルの全体構成

Fig. 2 Data structure of compact file.

- parameter × NP

parameter

以下のいずれかとする．

- parameter，parameter \0
- parameterSet，parameterSet No.

4.2.2.6 parameterSet テーブルの内容

- parameterSet 数 NS
- parameterSet へのポインタ × NS
- parameterSet × NS

parameterSet

parameter \0

5. 評価システムの開発

これまでの検討結果に基づき，カメラやレコーダなどの監視コンポーネントに搭載し，制御 PC を使用せずコンポーネント内部でのイベント処理を実現する軽量ルールエンジンの評価システムを，文献 [19] で開発したシステムを拡張する形で開発した．

5.1 システム構成

図 3 に軽量ルールエンジンを適用した映像監視システムの評価システム構成について示す．

評価システムでは，新たにネットワークカメラに搭載する軽量ルールエンジンの機能をターゲットノード上に実装した．ターゲットノードには，将来のネットワークカメラへの搭載を想定して，一般的なネットワークカメラと同等の CPU 性能を持ち，組み込み Linux 環境が動作する CPU ボードを選択した．**表 3** にターゲットノードの仕様について示す．

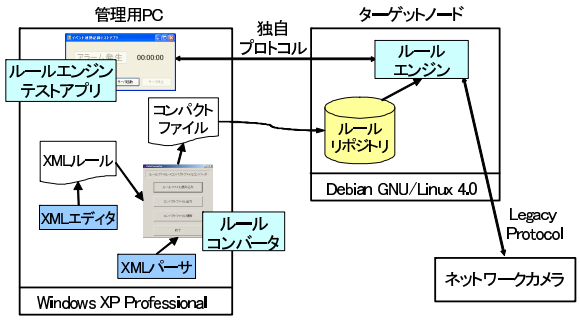

図 3 評価システム構成

Fig. 3 Prototype system configuration.

表 3 ターゲットノードの仕様

Table 3 Specification of target node.

プロセッサ	Cirrus Logic EP9315（ARM9）
システムクロック	200 MHz
SDRAM	64 MB
OS	Debian GNU/Linux 4.0
カーネルバージョン	2.6.12.3-a9-11

表 4 管理用 PC の仕様
Table 4 Specification of management PC.

プロセッサ	Intel Pentium M
システムクロック	1,400 MHz
RAM	1,024 MB
OS	Windows XP Professional SP3

また，システム評価に必要となる管理用 PC として Windows XP Professional が動作する PC を準備し，これらの装置を 100BASE-TX ネットワークで接続する構成とした．**表 4** に管理用 PC の仕様について示す．

本システムの特長について以下に示す．

- ルールを XML で記述しているため，市販またはオープンソースのツール（XML エディタや XML パーサ）を利用でき，開発コストを削減できる．
- 監視コンポーネントはリソース（CPU，メモリ）が限られているため，XML エディタによるルールの作成や XML パーサによるルールの解析は管理用 PC で行う．
- ルールの実行部分（ルールエンジン）は監視コンポーネントで動作し，汎用部とデバイス依存部から構成される．

5.2 ルール処理の流れ

本システムでは，XML エディタや XML パーサなどの XML 処理系を管理用 PC に搭載し，XML エディタを用いた動作ルールの編集や XML パーサによるルールの解析は管理用 PC で行い，その解析結果を各監視コンポーネントに配布する．

また，動作ルールの監視コンポーネントへの格納形式であるが，システム設定時に管理用 PC でルールを XML パーサで解析して得られた DOM 形式のデータを，動作ルールに特化した独自のコンパクト形式に変換し，監視コンポーネント上のルールリポジトリ上に格納しておく．システム運用時に監視コンポーネント上で動作するルールエンジンでは，ルールリポジトリ内にコンパクト形式で格納されたルールを読み出して使用する．

5.3 ルールコンバータ

管理用 PC で動作し，XML 形式で記述された動作ルールを読み込み，XML パーサによるスキーマチェックを行い DOM 形式で内部メモリに展開後，コンパクト形式へ変換・出力する機能を持つルールコンバータを Windows アプリケーションとして開発した．XML パーサとしては，Microsoft XML Core Services 4.0 SP2 [20] を使用した．

5.4 ルールエンジン

コンパクト形式のルールファイルを読み込み，ルールに従いイベント処理を行う機能を持つルールエンジンを，ターゲットノード上に実装した．軽量ルールエンジンのモジュール構成を**図 4** に示す．

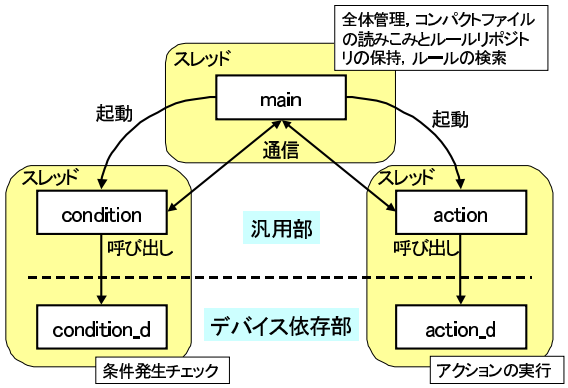

図 4 軽量ルールエンジンのモジュール構成
Fig. 4 Module configuration of Light-Weight Rule Engine.

図 4 に示したとおり，軽量ルールエンジンの内部は「main」，「condition」，「action」の 3 スレッドで構成される．以下，軽量ルールエンジンの処理フローについて示す．

(1) 初期化処理

main スレッド ルールリポジトリからコンパクトファイルを読み込み内部に展開し，condition スレッド，action スレッドの起動を行う．

(2) イベント処理

condition スレッド ルールに従い，状態変化をポーリングし，イベント発生を受け，main スレッドに通知する．

main スレッド メッセージを受信し，ルールリポジトリを検索する．対応したルールが見つかった場合，action スレッドに通知する．

action スレッド メッセージを受信し，対応する action を実行する．

初期化処理はルールエンジンの起動時に 1 回実行すればよいため，リアルタイム要求は低いが，イベント処理はリアルタイム要求が高い．また，アクションを実行中でも別のイベントが発生しうるため，これらの 3 つの機能をそれぞれ別のスレッドで実行する．

また，ルール検索処理を行う main スレッドはデバイス非依存であるが，条件チェックを行う condition スレッドとアクション実行を行う action スレッドは，用途に応じてデバイス非依存の部分が存在する．評価システムでは，3 章で定義したシナリオ 1 とシナリオ 2 の 2 種類のシナリオに対応した実行モジュールを実装した．

5.5 ルールエンジンテストアプリ

ルールエンジンの評価を目的として，管理用 PC 上で動作し，3 章で定義したシナリオ 1 とシナリオ 2 の 2 種類のシナリオに対応したイベントを発生させるルールエンジンテストアプリを Windows アプリケーションとして開発した．

6. 評価

評価システムを用いて，ルールコンバータによる XML ルールの圧縮効率，ルールエンジンのイベント処理性能，ルールエンジンのメモリ使用量についてそれぞれ評価を行った．

評価において，3 章で定義したシナリオを性能評価の観点から分析した結果，シナリオ 2 がシナリオ 1 を含むと判断し，基本的にシナリオ 2 を使用する方針とした．

6.1 XML ルールのコンパクト化効率の評価

XML 形式で記述したルールファイルをルールコンバータでコンパクト化した際の圧縮効率について評価した．

シナリオ 2 を想定し，無線装置からの帯域情報を定期的に取得し，ルールに従ってネットワークカメラの符号化パラメータ（フレームレート，圧縮率，解像度）を変更するルールを定義した．

評価に際して，ルール数を 1 から 10 まで 1 ずつ増やし，ルールコンバータによる XML ルールのコンパクト化効率を評価した．参考用として，同じ XML データに対して gzip [6] および XMill–0.8 [21] を用いてデータ圧縮を行った．その結果を**図 5** に示す．

図 5 において，データ圧縮方式である gzip や XMill と比較して，圧縮率は及ばないものの，平均 56.4%とコンパクト化の効果を確認できた．

6.2 ルールエンジンのイベント処理性能の評価

評価システムを用いて，ルールエンジンのイベント処理性能の評価を行った．具体的には，シナリオ 2 を用いて，無線回線はモバイル WiMAX [22] を想定して，6.1 節で作成したルールの中から最も実用的なルール（ルール数 4）を選択して性能評価を実施した．なお，ネットワークカメラの符号化方式は JPEG とし，取得した帯域情報をルール処理し，その結果に従ってフレームレート，圧縮率，解像度を変更するルールとした．**表 5** に評価で使用したルールの概要について示す．

評価システムを用いて，表 5 で定義した動作ルールを用いて，ルールエンジンに実装したログ出力機能を用いて，以下の処理時間を 10 分間測定した．なお，ルールエンジンテストアプリとの帯域情報の取得周期は 1 秒に設定した．

イベント処理時間 ルールエンジンテストアプリと定期的に通信し，伝送可能帯域を取得する時間．

ルール処理時間 取得した帯域とルールの比較により，条件が適合した際にイベントを発行し，動作ルールに適合したアクションを検索する時間．

アクション処理時間 ルール処理結果に基づき適切なルール処理を行う時間．今回は対象ネットワークカメラに対して適切なパラメータを設定する時間．

ルールエンジンのイベント処理性能の評価結果を**表 6** に示す．

表 6 において，イベント処理性能の評価結果が，平均 7.857 msec という結果が確認できた．

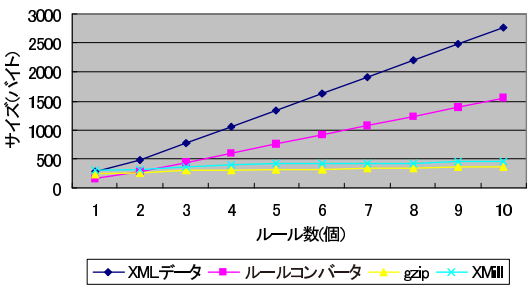

図 5 XML ルールのコンパクト化効率の評価結果

Fig. 5 Evaluation results of compression efficiency of XML rule.

6.3 ルールエンジンのメモリ使用量の評価

評価システムを用いて，ルールエンジンのメモリ使用量の評価を実施した．具体的には，6.1 節で使用したルールを使用し，ルール数を 1 から 10 まで 1 ずつ増やす形で，ルールエンジンの起動前と動作中の`/proc/meminfo` ファイルの内容を解析する形でメモリ使用量を測定した．

一般的には，メモリ使用量 $MemUse$ は，全メモリ $MemTotal$ から空きメモリ $MemFree$ を引くことにより算出できるが，ターゲットノード上で動作する Linux-2.6 では，バッファ $Buffers$ やキャッシュ $Cached$ も使用メモリとして加算されているため，`/proc/meminfo` ファイルの内容に対して，以下の数式を用いて算出した．

$$MemUse = MemTotal - MemFree - Buffers - Cached \quad (1)$$

表 5 評価用ルールの概要

Table 5 Overview of the rules for evaluation.

取得帯域 (Mbps)	フレームレート (fps)	圧縮率	解像度
1.33～	15	1/20	QVGA
0.67～1.33	7.5	1/20	QVGA
0.33～0.67	3	1/20	QVGA
～0.33	1	1/20	QVGA

表 6 イベント処理性能の評価結果

Table 6 Evaluation result of the event processing performance.

評価対象	平均値 (msec)	最大値 (msec)	最小値 (msec)
イベント処理	2.918	29.072	1.569
ルール処理	2.324	3.811	2.193
アクション処理	2.614	11.332	2.253
合計	7.857	33.969	6.130

表 7 メモリ使用量の評価結果
Table 7 Evaluation result of memory usage.

評価対象	平均値（KB）	最大値（KB）	最小値（KB）
ルールエンジン	148.8	160	132
libxml2	186	208	152

表 8 CPU 別でのイベント処理性能の評価結果
Table 8 Evaluation result of the event processing performance according to CPU.

評価対象	初期化処理（msec）	イベント処理（msec）
Pentium4（3.8 GHz）	10.1	1.575
ARM（200 MHz）	31.9	1.475

ルールエンジンの起動前と動作中の`/proc/meminfo`から算出したメモリ使用量を比較し，実際のメモリ使用量を算出した．また，比較用に，同じ XML データを標準的な XML 処理ソフトウェアの 1 つである libxml2–2.6.27 [23] を用いて DOM 形式で読み込んだときのメモリ使用量も合わせて算出した．

ルールエンジンのメモリ使用量の評価結果を**表 7** に示す．

メモリ使用量の評価において，ルール数を 1 から 10 に変更した場合でも，メモリ使用量に特に大きな変化は見られなかった．

6.4 ルールエンジンのスケーラビリティの評価

ルールエンジンのスケーラビリティの評価を行うために，評価システムで開発したルールエンジンのデバイス依存部である「condition_d」と「action_d」を改修する形で，シナリオ 1 とシナリオ 2 の両方に対応するルールエンジンの実行モジュールを実装した．次に，シナリオ 1（ルール数 2）とシナリオ 2（ルール数 4）を組み合わせたルールを作成し，6.2 節と同じ要領で，ルール処理におけるイベント処理性能の評価を行った．

その結果，表 6 とほぼ同等の結果が得られた．

さらに，制御 PC と組み込み Linux 環境での CPU 別でのイベント処理性能の比較を目的として，PC（Pentium4 3.8 GHz）の Linux 上に評価用の軽量ルールエンジン実装し，実運用を想定したシナリオ（ルール数 3：4 つの OR 条件の AND を条件とするルール，4 つの AND 条件の OR を条件とするルール，5 つの AND 条件の THEN を条件とするルールから構成）を用いてイベント処理機能の性能評価を実施した．評価において，初期化処理は 1 回，イベント処理は 4 回測定し，その平均値を求めた．

その結果を**表 8** に示す．

この結果は，組み込み環境では，ファイル処理などが必要となる初期化処理は 2 倍以上の時間がかかるが，リアルタイムでの処理が必要なイベント処理は，イベントが輻輳していなければ，PC と同程度の性能が確保できていることを確認できた．

7. 考察

7.1 XML ルールのコンパクト化効率の評価に対する考察

6.1 節の評価結果について考察した結果，データ圧縮方式である gzip や XMill と比較して，圧縮率は及ばないものの，平均 56.4%とコンパクト化の効果を確認できた．

また，一般的な XML データをそのまま読み込んだ場合と比べて，コンパクト化した XML データはルールエンジンでそのまま読み込んで使用することができるため，組み込み機器上での動作において，メモリ低減効果が期待できる．

今回，評価は行っていないが，一般的に組み込み機器ではファイル処理がボトルネックになることが多く，その点においてもコンパクト化の効果が期待できる．

また，参考用に実施した gzip や XMill は，XML データを使用する際にいったんデータを伸張する必要があり，伸張後は DOM 形式で扱う必要があるため，使用メモリの観点からも，本論文で対象としているルールエンジンへの適用は難しくなる．

7.2 ルールエンジンのイベント処理性能の評価に対する考察

3 章で設定した性能要件をふまえて，6.2 節の評価結果について考察した結果，イベント処理の目標応答性能 33 msec に対して，評価結果が 7.857 msec となっており，目標性能を達成していることを確認できた．

この結果は，いったんコンパクトファイルをメモリ内に読み込んでしまえば，CPU 性能の低い組み込み機器においても，イベント処理の性能要求を満たすことが可能な方式と考える．

また，今回の評価では，ルールエンジンの入力となるイベントは 1 種類を前提条件として，イベント処理の性能評価を実施した．現状は目標値を十分満たしているが，実際の監視コンポーネントへの搭載を前提とした場合，形式の異なる複数のイベントに対応することが必要となり，イベント処理において輻輳が発生する確率が高くなることが予想される．現状，ルールエンジンではスレッド処理により並行動作に対応しているが，この点について考慮していく必要がある．

7.3 ルールエンジンのメモリ使用量の評価に対する考察

3 章で設定したメモリ要件をふまえて，6.3 節の評価結果について考察した結果，メモリの削減効果目標 20%に対し，評価結果が 20%となり目標性能を達成していることを確認できた．また，評価において，ルール数を 1 から 10 に増やして評価を行ったが，メモリ使用量は特に大きな変化

は見られなかった．

この結果は，XML パーサを動作させるとある程度のメモリ容量が必要になるが，XML パーサを動作させないことにより，メモリ使用量の削減が期待できるという仮説を実証できていると考える．

また，ルール数を 1 から 10 に増やしても，メモリ使用量とはオーダが異なるため，必要となるメモリ使用量はほとんど変わらないことが確認できた．3 章で設定したシナリオでは，想定しているルール数としてシナリオ 1 が 1〜2，シナリオ 2 が 1〜10 であり，今回設定したシナリオの範囲内では，メモリ使用量の削減効果を確認できた．

7.4 ルールエンジンのスケーラビリティの評価に対する考察

6.4 節の評価結果を考察した結果，開発したルールエンジンは，動作させたいシナリオにあわせて，デバイス依存部である「condition_d」と「action_d」を改修することにより，異なるシナリオに対応できることを確認した．

関連研究である文献 [5] のルールエンジンでは，プラグインという形で Event や Action の拡張機能を実装することが可能であるが，本ルールエンジンでは，デバイス依存部のソースコード改修と実行モジュールの再ビルドが必要となる．一般的には，いったん監視コンポーネントにルールエンジンを組み込んでしまえば，運用時はルールの変更で対応できる場合が多く，運用上の不都合は発生しないと考える．

また，複数シナリオに対するルールエンジンのイベント処理性能については，評価環境では，ルールエンジンのトリガとなる Event が，シナリオ 1 では「特定のアラームイベント発生」，シナリオ 2 では「帯域情報の取得周期は 1 秒」に設定されていたため，複数シナリオの動作時でもルール処理におけるイベントの輻輳が発生しなかったためと考える．

表 6 のとおり，評価システムにおけるルールエンジンのイベント処理性能は平均 7.857 msec であることを考慮すると，シナリオの種類に関係なく，ルールエンジンのイベント発生周期を 10 msec 以上に調整できれば，イベントの輻輳を回避することができると考える．表 8 の結果もあわせて考察すると，イベントの輻輳が発生しなければ，自コンポーネントに対するルール処理の移行が可能であると考える．

8. おわりに

一般的な映像監視システムでは，監視カメラやセンサなどから発生したすべてのイベントを制御 PC で処理することになるため，システムコストが高くなるとともにイベントが集中したときのシステムのレスポンス低下が課題となっていた．そこで本研究では，カメラやレコーダなどの監視コンポーネントに搭載し，制御 PC を使用せずにコンポーネント内部でイベント処理を実現することを目的とした軽量ルールエンジンを開発した．

組み込み Linux 環境上に軽量ルールエンジンを実装し，その処理性能およびメモリ使用量について評価を行った．評価実験の結果，組み込み Linux 向けに開発した軽量ルールエンジンは，XML 形式で記述されたルールを独自形式にコンパクト化することにより，組み込み向け CPU において，イベント処理性能およびメモリ使用量について性能要件を達成していることが確認できた．

また，今回ターゲットノードの CPU として採用した ARM 環境は様々な SoC における内蔵 CPU として採用されており，本方式は監視コンポーネント以外にも実時間でのイベント処理が必要な様々な組み込み機器に広く適用が可能である．

参考文献

[1] 矢野経済研究所：2008〜09 年版 ビジュアル・コミュニケーションシステム市場 Vol.1 ネットワークカメラ編 (2007).

[2] 山田陽一，山口政巳，本玉靖和：ユビキタスセンサネットワークにおける映像監視システム，沖テクニカルレビュー，Vol.72, No.4, pp.44–47 (2005).

[3] JBoss Drools, available from ⟨http://www.jboss.org/drools/⟩.

[4] OpenRules, available from ⟨http://openrules.com/⟩.

[5] 寺田 努，宮前雅一，山下雅史：Wearable Toolkit：その場プログラミング環境実現のためのイベント駆動型ルール処理エンジンおよび関連ツール，情報処理学会論文誌，Vol.50, No.6, pp.1587–1597 (2009).

[6] The gzip home page, available from ⟨http://www.gzip.org/⟩.

[7] Liefke, H. and Suciu, D.: XMill: An Efficient Compressor for XML Data, *Proc. ACM SIGMOD 2000*, pp.153–164 (2000).

[8] 東原正智，田島敬史：問い合わせ処理に適した XML データ圧縮形式に関する研究，電子情報通信学会 第 15 回データ工学ワークショップ DEWS2004，4-C-04 (2004).

[9] Tolani, P.M. and Haritsa, J.R.: XGrind: A Query-friendly XML Compressor, *Proc. IEEE ICDE 2002*, pp.225–234 (2002).

[10] Min, J.-K., Park, M.-J. and Chung, C.-W.: XPress: A Queriable Compression for XML Data, *Proc. ACM SIGMOD 2003*, pp.122–133 (2003).

[11] Peintner, D., Kosch, H. and Heuer, J.: EFFICIENT XML INTERCHANGE FOR RICH INTERNET APPLICATIONS, *Proc. IEEE ICME 2009*, pp.149–152 (2009).

[12] Efficient XML Interchange (EXI) Format 1.0, available from ⟨http://www.w3.org/TR/exi/⟩.

[13] 吉田 茂，中島 哲，小田切淳一，伊藤秀一：データ処理性能を改善する XML データのコンパクト化法の開発，電子情報通信学会論文誌，Vol.J89-D, No.4, pp.767–777 (2006).

[14] 小林亜令，村松茂樹，西山 智：XEUS：携帯電話向け XML データ符号化方式，情報処理学会論文誌，Vol.50, No.1, pp.209–221 (2009).

[15] 中島 哲，小田切純一，井谷宣子，吉田 茂：XML 高速

処理技術 SPlitDOM の機能拡張と Web アプリケーションへの適用評価，電子情報通信学会 第 15 回データ工学ワークショップ DEWS2004，I-5-05 (2004).
[16] Widom, J. and Ceri, S.: *Active Database Systems: Triggers and Rules for Advanced Database Processing*, Morgan Kaufmann (1995).
[17] Paton, N.W.: *Active Rules in Database Systems*, Springer-Verlag (1998).
[18] W3C XML Schema, available from ⟨http://www.w3.org/XML/Schema⟩.
[19] 阿倍博信，中島宏一，峯村治実：XML ルール圧縮による監視コンポーネント向け軽量ルールエンジン，第 8 回情報科学技術フォーラム FIT2009，M-001 (2009).
[20] MSXML 4.0 SP2 Release Notes, available from ⟨http://download.microsoft.com/download/9/6/5/9657c01e-107f-409c-baac-7d249561629c/MSXML4SP_RelNote.htm⟩.
[21] XMill Project Top Page, available from ⟨http://en.sourceforge.jp/projects/sfnet_xmill/⟩.
[22] IEEE 802.16e-2005: IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Institute of Electrical and Electronics Engineers (2006).
[23] The XML C parser and toolkit of Gnome, available from ⟨http://xmlsoft.org/⟩.

阿倍 博信 （正会員）

昭和 63 年慶應義塾大学理工学部計測工学科卒業，平成 2 年同大学大学院理工学研究科修士課程修了，同年三菱電機株式会社入社．以来，グループウェアシステム，マルチメディア応用システムの研究開発に従事．平成 17 年慶應義塾大学大学院理工学研究科後期博士課程修了．博士（工学）．電子情報通信学会，日本ソフトウェア科学会，映像情報メディア学会，画像電子学会，教育システム情報学会各会員．

中島 宏一

昭和 59 年早稲田大学理工学部電気工学科卒業，同年三菱電機株式会社入社．以来，テレビ会議システム，映像監視システム等のマルチメディア応用に関わる研究開発に従事．

峯村 治実

昭和 59 年東京大学工学部電気電子工学科卒業，昭和 61 年同大学大学院工学系研究科（電子工学）修士課程修了，同年三菱電機株式会社入社．以来，データベースマシン，RAID，Java アプリケーションプラットフォーム，機器遠隔監視・制御技術等の研究開発に従事．