Frontmatter

2 Editors

3 S. Shin SIGAPP FY'10 Annual Report

5 SAC 2010 Overview

Invited Paper

7 Y. Cho J. Jung Virtual Protocol Stack for WSN Simulators

Research Articles


27 L. Cesari R. Pugliese F. Tiezzi A Tool for Rapid Development of WS-BPEL Applications

Mission
To further the interests of the computing professionals engaged in the development of new computing applications and to transfer the capabilities of computing technology to new problem domains.

Officers

Chair  
Sung Shin  
South Dakota State University, USA

Vice Chair  
Richard Chbeir  
Bourgogne University, Dijon, France

Secretary  
W. Eric Wong  
University of Texas, USA

Treasurer  
Lorie Liebrock  
New Mexico Institute of Mining and Technology, USA

Web Master  
Hisham Haddad  
Kennesaw State University, USA

ACM Program Coordinator  
Irene Frawley  
ACM HQ

Notice to Contributing Authors
By submitting your article for distribution in this Special Interest Group publication, you hereby grant to ACM the following non-exclusive, perpetual, worldwide rights.

• to publish in print on condition of acceptance by the editor
• to digitize and post your article in the electronic version of this publication
• to include the article in the ACM Digital Library
• to allow users to copy and distribute the article for noncommercial, educational or research purposes

However, as a contributing author, you retain copyright to your article and ACM will make every effort to refer requests for commercial use directly to you.

Status Update
The main event that took place within SIGAPP for this year was the Symposium on Applied Computing (SAC) in Switzerland after taking place in Honolulu, Hawaii for 2009. This year’s SAC was very successful. More details about SAC will follow in the next section. We also supported several additional conferences with in-cooperation status, and will continue supporting additional conferences in the coming year.

SIGAPP’s web page has been redesigned by Hisham Haddad, Web Master of SIGAPP in April. A brand new SIGAPP logo has been selected from a logo competition and will be used for the SIGAPP web page and official publications such as the proceedings of SAC and Applied Computing Review.

I’m pleased to announce the reissuing of ACR (Applied Computing Review). As you know, ACR is the newsletter of the ACM SIGAPP. It hasn’t been published since 2002 and has been strongly desired by
the members of SIGAPP. We’re introducing it semi-annually as an electronic version only. Once the format has stabilized, we’ll begin publishing quarterly electronically and in print. Ultimately, we want ACR to appear in the SCI (Science Citation Index). ACR contains invited papers from world-renowned researchers and selected papers presented by prominent researchers and professionals in the Symposium on Applied Computing 2010 in Sierre, Switzerland. The selected papers have been expanded, revised, and peer-reviewed again for publishing in ACR. We hope that ACR will serve as a platform for many new and promising ideas in the many fields of applied computing. As you know, it is strongly related to nearly every area of computer science, and we feel an obligation to serve you as best we can. The papers in this issue of ACR represent the current applied computing research trends. These authors truly contribute to the state of the art in applied computing.

The Student Travel Award Program continues to be successful in assisting SIGAPP student members in attending conferences sponsored by or in-cooperation with SIGAPP. 24 students were granted awards to attend SAC 2010, representing 14 countries. This was a bit less than last year, but we supported everyone who applied for Travel Award Program. The allocated budget for these awards increased compared to last year. We also implemented a Developing Countries Travel Award for researchers from developing countries who would otherwise have difficulty attending the SAC conference. For 2010, this award was used exclusively for students from developing countries, but in 2011 and beyond we hope to support faculty-level researchers from such countries.

SIGAPP continues to have a stable membership, and its strength and uniqueness among ACM SIGs continues to be an opportunity for scientific diversity and crosscutting multiple disciplines within the ACM community. To encourage this, SIGAPP is continuing to accept proposals for new tracks this year. The officers look forward to working with the ACM SIG Governing Board to further develop SIGAPP by increasing membership and developing a new journal on applied computing. They also appreciate the opportunity to support the programs of SIGAPP since they have provided a springboard for further technical efforts and have done a great service to many technical communities.

New SIGAPP logo

For the first time ever, SIGAPP has an official logo. Designers from around the world sent in their entries as part of a logo competition, chaired by Dr. Richard Chbeir. After five months in the selection process, the final logo design was chosen by the logo competition task force, and Virginia Sprang became the victor. Congratulations!

Next Issue

The planned release for the next issue of ACR is February 2011.
SAC 2010 Overview
SAC Steering Committee

The 25th Annual edition of SAC has marked another successful event for the Symposium on Applied Computing. This international gathering attracted over 490 attendees from over 70 counties. It was hosted and held on the campus of University of Applied Sciences Western Switzerland (HES-SO) in Sierre, Switzerland, March 2010. On Monday, the tutorials program offered 8 tutorials and attracted over 60 attendees. The program included coffee breaks and a social luncheon that took place in a historic restaurant located near the campus. The four-day technical program included over 340 presentations from forty tracks covering a wide range of topics on applied computing. The successful posters programs attracted over 80 posters that were presented over two sessions on Wednesday. Thanks to a great organizing committee, it was extremely successful. In all, 1353 papers were submitted, the most in the history of the SAC, and the final acceptance rate for SAC 2010 was 26.9% for the overall track.

The Tuesday and Thursday keynote addresses were well attended and received. On Tuesday, Professor Bertrand Meyer, Professor of Software Engineering at the Swiss Federal Institute of Technology, talked about the future of programming and shared his vision of how programming will evolve over the next 10 years. He also discussed how current and new techniques can help software developers cope with society’s increasing software demands, both quantitative and qualitative. The address was well received and followed by an active Q&A session. On Thursday, Professor Willy Zwaenepoel, Professor and Dean of the School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland, talked about the dilemma of research publications and the impact of published work in real systems. Specifically, the contradiction between the complexity of published work and simplicity of ideas to have positive impact in real systems. He presented examples of ideas that were successfully transferred to practice, and some ideas on how one can improve the situation. The address was well received by the audience and followed by an active Q&A session.
A number of business meetings were held during SAC 2010, mainly the SIGAPP annual meeting to discuss the current and future directions of the SIG, the SAC organization meeting where proposals for hosting future SAC meetings are presented, and the Track Chairs business meeting where feedback and planning for future SAC tracks are discussed. The social program included daily lunches and coffee breaks provided on site, allowing attendees to gather, converse, and network. On Tuesday, SIGAPP and the host institution hosted a reception. The main social event was the Banquet that was held at Center Le Régent Crans-Montana. The event included live entertainment and the Awards Ceremony. SAC Steering Committee encourages the readers to check SAC 2010 website for more information about the conference events, best paper awards, and some pictures of the event. The committee also encourages the readers to participate in SAC 2011 in Taichung, Taiwan, March 2011. We hope to see you there.

**SAC Summary Info**

1. SAC 2010 Awards:
   1. Distinguished Service to SAC:
      Sung Shin
   2. Outstanding Service to SIGAPP:
      Barrett R. Bryant
   3. Student Travel Awards: 24 awards granted, totaling $12,671.15

2. New tracks in SAC 2010:
   1. Bioinformatics
   2. Computer Forensics
   3. Power-Aware Designed Optimization
   4. Privacy on the Web
   5. Applications of Evolutionary Computing

SAC 2011 will be held in Taichung, Taiwan, March 21-25, 2011, and will be hosted by the Tunghai University, Taichung, Taiwan. The web site has further details such as the symposium committee, technical tracks, and track chairs. [http://www.acm.org/conferences/sac/sac2011/](http://www.acm.org/conferences/sac/sac2011/)

SAC 2012 is being planned for Italy which has been selected from three SAC local host proposals: Korea, Italy, and Portugal.
Virtual Protocol Stack for WSN Simulators

Yookun Cho  
Seoul National University, Korea  
cho@os.snu.ac.kr

Jinman Jung  
Seoul National University, Korea  
jmjung@os.snu.ac.kr

ABSTRACT

In the most recent years, many kinds of simulation packages have been developed for wireless sensor networks. Network developers, engineers, and researchers are frequently using the network simulators for evaluating performance and efficiency of their network protocols. However, the existing wireless network simulators have some drawbacks such as difficulties in programming, lengthy compile time, hard to understand simulator engine, and their reliability issues. In this paper, we present the Virtual Protocol Stack (VPS) for wireless network simulators. The VPS provides two important features; 1) a unified programming interface for developing network protocols, and 2) separation of simulator engine and the network protocols. By using the VPS, developers no longer concern about understanding different simulator engines. By doing so, developers can share their network protocol modules with each other. We implemented and adapted VPS on existing well-known simulation packages, and we observed that VPS well adapted to the different wireless network simulators.

Categories and Subject Descriptors

I.6.7 [Simulation and Modeling]: Simulation Support Systems—Environments

General Terms

Performance, Measurement

Keywords

Simulation, Wireless Sensor Networks

1. INTRODUCTION

In most recent years, wireless sensor networks have been one of the hot topics in the field of computer science and engineering. Many network protocols, system-level techniques, huge number of hardware and software methods were developed to provide better performance in actual sensing environments [1]. They have been used in many different environments such as academies [16], forest [11], buildings, and even harsh environments [8].

Typically, the sensor networks are composed of many wireless sensor nodes deployed in the remote region to sense events. Each sensor node performs sensing, computing, and multi-hop wireless communication between nodes. The nodes are very resource-constrained to maximize cost-efficiency of the whole networks [7]. For example, Berkeley’s Mica series consist of 4-kB RAM, 8 bit CPU, radio transmission module, and limited alkaline or lithium batteries [4].

In wireless sensor networks, energy-efficiency is the most important factor because the sensor nodes have limited batteries. Huge number of researches have been made to provide more energy-efficient communication, sleep scheduling, and even system-level software techniques. On developing phase of new mechanisms, it is crucial to measure their performance and energy-efficiency in the whole sensor networks. However, using the actual sensor networks require significant cost to buy, deploy, and manage them. This is the reason why the developers need some instruments to simulate the whole wireless sensor networks. By using simulation packages, the developers can minimize the time and cost consumed in the performance evaluation. In addition, simulation packages can generate a lot of valuable information which is hard to get in the actual sensor networks.

Many network simulators have been developed to test and evaluate performance of new mechanisms in the wireless sensor networks such as SENSE [3], EmStar [5], GloMoSim [6], TOSSIM [9], ns-2 [12], OMNeT++ [13], OPNET [14], QualNet [15], and SensorMaker [18]. However, they still have some drawbacks such as difficulties in programming, lengthy compile time, hard to understand simulator engine, and their reliability issues. Some of them are hard to use for various communication protocols [9], and some of them are not appropriate to measure and evaluate network routing and clustering protocols. In addition, some simulators are too complex to understand their internal structures and programming interfaces [12].

In this paper, we present the Virtual Protocol Stack (VPS) for network simulations in wireless sensor networks. VPS is mainly focused on the following two features; 1) providing a unified programming interface for developing network protocols, and 2) separation of simulator engine and the network protocols. This approach is much similar to existing VFS, which is an architecture for multiple file systems. By us-
ing the VPS, developers do not need to understand multiple different simulator engines. When developer develops a network protocol binary module according to the VPS, it can be simulated for multiple wireless sensor network simulators without any modifications. We implemented and adapted VPS on existing well-known simulation packages, and we observed that VPS well adapted to the different wireless network simulators.

The rest of this paper is organized as follows. In Section 2, we present several previous works on the existing simulators for wireless sensor networks. Section 3 presents the internal structures of the VPS. Section 4 presents the implementation and usage of the VPS, and Section 5 evaluates performance and reliability of the VPS in the large-scale wireless sensor networks. Finally, some conclusions and future work are given in Section 6.

2. RELATED WORKS

In this section, we present several previous simulators related to our proposed virtual protocol stack. Many research efforts were made on developing simulation and instrumentation toolkits for wireless sensor networks [2, 3, 5, 9, 6, 12, 13, 14, 15, 17, 18].

The ns-2 [12] is the famous simulator for both wired and wireless networks. It was developed by LBNL, UC Berkeley, CMU, and many other research institutes, and it has been used for many kinds of previous works related to the network protocols. However, ns-2 is very hard to program because it uses both OTcl (object-oriented tool command language) and C++, and the internal structures are too complex to understand thoroughly. For this reason, many efforts have been made to implement easier simulation packages than the ns-2.

The EmStar [5] is a wireless network simulator which was developed by UCLA in 2004. It adapts the event-driven architecture, and supports graphic interface on the Linux’s X-window. The EmStar was designed for detecting faults and problems between sensor nodes, and there are many documents and tutorials on their website. However, it also has few functions to measure the performance of the sensor nodes, and it is used only small number of researches.

The TOSSIM [9] is a well-known simulator for applications running on TinyOS [10]. It was developed by UC Berkeley’s TinyOS project team. The main goal of the TOSSIM was to provide debugging method for the TinyOS and its applications. However, it can simulate only the sensor applications ported to the i386 architecture. It was not precise as much as AVRORA, and it cannot be used for other sensor operating systems and their applications.

Yi et al. proposed SensorMaker [18] for scalable and fine-grained simulation in wireless sensor networks. It provides an easy way to program communication protocols, and it has been used for many researches including routing, clustering, mobile checkpointing, software updating protocols. However, the SensorMaker does not provide component-based implementation, and the developers should understand their whole internal structures to implement some network protocols.

The GloMoSim [6] is a mobile network simulation library developed at the UCLA. It was designed as a set of library modules and developed using PARSEC, a C-based parallel simulation language. The GloMoSim is extensible, composable and suitable for wireless ad-hoc and sensor networks. However, the protocol developers should understand the entire structures of the GloMoSim and should be familiar with PARSEC.

The OMNeT++ [13] is a discrete event simulation environment. It provides the basic machinery and tools to conduct simulations. The OMNeT++ adopted the component-based architecture. In the OMNeT++, the network simulation environments, called as models, are assembled from several reusable modules based on a NED (NETwork Description) language. Therefore, protocol developers should be familiar with the concept of models, components, and the NED language to develop their protocols in the OMNeT++. The OPNET [14] stands for OPtimized Network Engineering Tools. The OPNET Modeler is based on a series of hierarchically-related editors. Among these editors, a process editor describes the behavior and functions of the modules. It uses a finite state machine to describe the protocols in detail. Each state of a process contains C/C++ code for specific controlling. For this reason, protocol developers should understand the process model to implement their protocols in OPNET. The QualNet [15] extends GloMoSim for the commercial use on both industries and academies. It provides many kinds of verified built-in network protocols on the simulation package. However, the process for adding new protocols to the QualNet is rather complex and contains some unnecessary procedures.

3. VPS: VIRTUAL PROTOCOL STACK

In this section, we present several requirements for network simulations. Then, we describe the design and implementation of the VPS in detail.

3.1 Requirements for Network Simulations

On developing new network protocols, it is crucial to measure their performance. For this, we need to use a network simulation package which provides various kinds of simulation aspects. In this paper, we focused on designing a better structure for network simulators enabling easier management of various network protocols. Several requirements considered are as follows.

- Minimizing the amount of source codes which are to be analyzed for implementing a new network protocol
- Minimizing complexity of internal structure of the simulator engine
- Providing independent development environment by removing dependencies between network protocols and simulator engine
- Providing easier programming interface for network protocol developers
- Providing fault-tolerance on simulator engine for unstable network protocols
- Supporting easier method for building a new network protocol
- Supporting run-time network protocol plug-in facility
3.2 Design of VPS

VPS stands for virtual protocol stack between protocols and simulator engine. Similar to middlewares between applications and operating systems, it provides an adaptation layer between protocols and the engine. By using the VPS, simulator developers can easily implement, manage, and fix their software components. Also, protocol developers do not need to understand whole structures of the simulator. The developers can easily implement various network protocols by using the generalized interface provided by the VPS.

3.2.1 Overview

Figure 1: Comparison of Existing Network Simulator and VPS-based Simulator

Figure 1 (a) shows a previous network simulator which is composed of the simulator engine and several protocol stacks. In this structure, we need to rebuild the whole simulator program codes even if we modify just one network protocol. However, in Figure 1 (b), the VPS-based simulator can manage multiple protocols via hot plug-in method. The network protocols can be implemented separately, and protocol developers do not need to consider what kind of network simulator is to be used. In addition, developers can share their network protocol modules with each other.

![Figure 1: Comparison of Existing Network Simulator and VPS-based Simulator](image)

3.2.2 VPS SAL

SAL is a simulator adaptation layer which provides adaptation for various simulation packages. This is much similar to the concept of VFS’s interface and Java VM. By using SAL, VPS can be easily ported to any programmable network simulators. In this study, we ported SAL to the existing wireless sensor network simulators and we will port it to other simulators in our future work. Table 1 shows some primitive APIs defined in SAL interface.

### Table 1: SAL Interface

<table>
<thead>
<tr>
<th>SAL Interface</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>i32 vpsi_init</code></td>
<td>VPSI_CONF *</td>
</tr>
<tr>
<td><code>i32 vpsi_release</code></td>
<td>void</td>
</tr>
<tr>
<td><code>i32 vpsi_enum_ptcinfo</code></td>
<td>ENUM_PTC_INFO *</td>
</tr>
<tr>
<td><code>i32 vpsi_get_ptcinfo</code></td>
<td>u32 ptcID,PTC_INFO *</td>
</tr>
<tr>
<td><code>i32 vpsi_set_ptc</code></td>
<td>u32 sessionID,u32 ptcID</td>
</tr>
<tr>
<td><code>i32 vpsi_insert_ptc</code></td>
<td>char *path</td>
</tr>
<tr>
<td><code>i32 vpsi_remove_ptc</code></td>
<td>u32 ptcID</td>
</tr>
<tr>
<td><code>u32 vpsi_create_session</code></td>
<td>Node *</td>
</tr>
<tr>
<td><code>i32 vpsi_destroy_session</code></td>
<td>u32 sessionID</td>
</tr>
<tr>
<td><code>Session * vpsi_get_session</code></td>
<td>u32 sessionID</td>
</tr>
<tr>
<td><code>i32 vpsi_send_packet</code></td>
<td>u32 sessionID,Packet *</td>
</tr>
<tr>
<td><code>i32 vpsi_recv_packet</code></td>
<td>u32 sessionID,Packet *</td>
</tr>
<tr>
<td><code>i32 vpsi_handle_event</code></td>
<td>u32 sessionID,Event *</td>
</tr>
</tbody>
</table>

3.2.3 Component Management Part

The component management part manages protocols as dynamically loadable components. This part is designed to allow easy addition and removal of customer protocols without recompilation of simulators using it. By doing so, it allows simulators to easily and automatically take advantage of the newest protocols.

3.2.4 Session Management Part

Figure 3(a) shows the session management part. For better scalable network simulations, it is crucial to minimize computing and memory resource usage. In VPS, we decided to share protocols for all nodes to reduce memory usage, and we allocated session1 information for each node to reduce computing resource usage. For example, if every node in a network use the same protocol, then the protocol will be shared for all nodes, and session information will be proportional to the number of nodes.

3.2.5 Wrapper Part

In general, custom protocols (or, in other words, third-party protocols) are not reliable because of their error-prone characteristic. For better reliability, as shown Figure 3(b), we use the Wrapper layer which verifies arguments, return values, and accessing addresses on the common data struc-

---

1In this paper, the session represents a data structure of a node’s information.
The component manager scans protocols, and performs plug-in for the scanned protocols. Then the simulator calls the `vpsi_create_session()` to create sessions for all nodes. After that, the VPS sets up connection between the simulator engine and the nodes in the **Setup phase**. Finally, the simulator engine can perform network simulation by calling the `vpsi_send_packet()`, `vpsi_recv_packet()` or `vpsi_handle_event()` in the **Handling phase**.

### 4. IMPLEMENTATION

We have implemented the **VPS** on the existing sensor network simulators such as **GloMoSim 1.2.3** (Windows version)\(^2\), and **CoSim**\(^2\).

We used C++ programming language to implement the **VPS** because most of the existing simulators are now using it. We used Microsoft Visual Studio 2005 as the main programming environment. Win32 API and MFC libraries are used to implement GUI and visualization of the wireless sensor networks.

Figure 5 shows the overall structures of the **VPS** and the **CoSim** which were implemented in this study. The **VPS** is based on the Win32 subsystems, and each component is implemented in the DLL file format to link and load them dynamically. Also, each protocol component follows the DLL format to provide hot plug-in features.

In general, the amount of time consumed for simulation is proportional to the number of nodes and the traffic volumes of the networks. To effectively minimize simulation time, some multicore-based powerful machines can be used to exploit parallelism of the network simulations. To increase the benefit of parallel execution, In Fig. 5, the event scheduler supports both the **event-driven** and the **multi-threaded** operations based on the **threads pool** structure.

Figure 6 shows the snapshot of the network simulator which adopted the **VPS** as the main interface for network protocols. In this figure, the fields **A** and **B** represent sensor nodes and the target 2-dimensional ground, respectively. The fields **C**, **D**, **E**, **F**, **G**, and **H** show interface for configuring options used in network simulations.

### 5. PERFORMANCE EVALUATION

In this section, we present performance of the **VPS** in several aspects. For this, we measured execution time, memory usage, and the overall performance by using **CoSim** in many different conditions.

Table 3 represents several parameters used in the simulation. We assumed the size of sensing field as 500m × 500m 2-dimensional ground, and the sink node is located at the center of the field. The distribution method of sensor nodes is done by **C++** standard pseudo-random generation functions, `srand()` and `rand()`. The number of nodes is 500, and each node transmits its own data packet to the sink node at every 3 seconds.

\(^2\) **CoSim** was developed by our research team.
1. vpsi_init
2. vpsi_set_protocol(sID, AODV)
3. vpsi_enum_info (&layerinfo)
   vpsi_enum_ptcinfo (&ptcinfo)
4. create protocol object
5. vpsi_create_session (&node)
6. create session

(a) Init phase
(b) Setup phase
(c) Handling phase

Figure 4: Inter-connection between VPS and Simulator Engine

Figure 6: Screenshot of Prototype Implementation

Table 3: Simulation Environment and Parameters used in this paper

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensing field</td>
<td>500m × 500m</td>
</tr>
<tr>
<td>Position of sink node</td>
<td>(250,250)</td>
</tr>
<tr>
<td>Distribution of sensor nodes</td>
<td>Random distribution</td>
</tr>
<tr>
<td>Radio transmission range</td>
<td>80m</td>
</tr>
<tr>
<td>Mobility of sensor nodes</td>
<td>Random (Max. 2m/s)</td>
</tr>
<tr>
<td>Number of sensor nodes</td>
<td>500</td>
</tr>
<tr>
<td>Protocols</td>
<td>Network: AODV</td>
</tr>
<tr>
<td></td>
<td>MAC: CSMA/CA</td>
</tr>
<tr>
<td></td>
<td>PHY: Nano-24 (CC2420)</td>
</tr>
</tbody>
</table>

Figure 7 shows the impact of VPS on the execution time of the network simulator. Original CoSim denotes the simulator without any VPS layer, and the others show the results when the Wrapper is disabled or enabled on the VPS-based CoSim. This result shows that the overhead of VPS layer including Wrapper is only 0.97 percent of the total execution time.

Table 4 represents the expected time for 4,000 rounds of simulation according to the number of sensor nodes. The results show that the execution time increases sub-exponentially by the number of sensor nodes. This is one of the characteristics of the network simulator. This result shows that the VPS has no sub-exponential order of magnitude on the execution overhead. In other words, the VPS still has reasonable overhead for simulating large-scale sensor networks such as 2,500 nodes.

6. CONCLUSIONS AND FUTURE WORK

In recent years, lots of simulation packages have been developed for simulating wireless networks. However, each of them has some drawbacks such as difficulties in programming, significant time consumption on building network protocols, hard to understand simulator engine, and their reliability issues. To alleviate such problems, we presented the Virtual Protocol Stack (VPS) for multiple network simula-
tors in wireless networks. The VPS provides abstraction of multiple simulator engines into the generalized programming interface. By using the VPS, developers can just program network protocols according to VPS, and then the protocols can be used for multiple network simulators. By doing so, developers can share their network protocol modules with each other. This will provide extra benefit on the performance evaluation and quality control of the network protocols being developed.

We are currently extending our work to port the VPS to several other network simulators such as ns-2, ns-3, and OMNeT++. In addition, a wizard-based development environment for protocol developers will be possible to provide an environment for protocol developers will be possible to provide extra benefit on the performance evaluation and quality control of the network protocols being developed.

We are currently extending our work to port the VPS to several other network simulators such as ns-2, ns-3, and OMNeT++. In addition, a wizard-based development environment for protocol developers will be possible to provide extra benefit on the performance evaluation and quality control of the network protocols being developed.

7. REFERENCES


About the authors:

Yookun Cho (M’91) received the B.E. degree from Seoul National University, Seoul, Korea, in 1971 and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, in 1978.

Since 1979, he has been with the School of Computer Science and Engineering, Seoul National University, where he is currently a Professor. In 1985, he was a Visiting Assistant Professor with the University of Minnesota, and from 2001 to 2002, he was the President of the Korea Information Science Society. He also served as the honorary conference chair for ACM SAC 2007. His research interests include operating systems, algorithms, system security, and fault-tolerant computing systems.

Jinman Jung received his B.S. degree in Computer Science from Seoul National University, Seoul, Korea, in 2008. He is currently a Ph.D. candidate of School of Computer Science and Engineering, Seoul National University, Seoul, Korea. His current research interests include operating systems, embedded systems and wireless networks.
Fraud Detection in Reputation Systems in e-Markets using Logistic Regression and Stepwise Optimization

Rafael Maranzato  
Universo Online Inc.  
Dept. of R & D  
São Paulo, SP, Brazil  
rmaranzato@uolinc.com

Adriano Pereira  
Federal Center for Technological Education of Minas Gerais (CEFET-MG)  
Dept. of Computer Science  
Belo Horizonte, MG, Brazil  
adriano@decom.cefetmg.br

Marden Neubert  
Universo Online Inc.  
Dept. of R & D  
São Paulo, SP, Brazil  
mneubert@uolinc.com

Alair Pereira do Lago  
Univ. of São Paulo - USP  
Dept. de Computação  
São Paulo, SP, Brazil  
alair@ime.usp.br

ABSTRACT
Reputation is the opinion of the public toward a person, a group of people, or an organization. Reputation systems are particularly important in e-markets, where they help buyers to decide whether to purchase a product or not. Since a higher reputation means more profit, some users try to deceive such systems to increase their reputation. E-markets should protect their reputation systems from attacks in order to maintain a sound environment. This work addresses the task of finding attempts to deceive reputation systems in e-markets. Our goal is to generate a list of users (sellers) ranked by the probability of fraud. Firstly we describe characteristics related to transactions that may indicate fraud evidence and they are expanded to the sellers. We describe results of a simple approach that ranks sellers by counting characteristics of fraud. Then we incorporate characteristics that cannot be used by the counting approach, and we apply logistic regression to both, improved and not improved. We use real data from a large Brazilian e-market to train and evaluate our methods and the improved set with logistic regression performs better, specially when we apply stepwise optimization. We validate our results with specialists of fraud detection in this market place. In the end, we increase by 112% the number of identified fraudsters against the reputation system. In terms of ranking, we reach 93% of average precision after specialists’ review in the list that uses Logistic Regression and Stepwise optimization. We also detect 55% of fraudsters with a precision of 100%.

Categories and Subject Descriptors
K.4.4 [Computers and Society]: Electronic Commerce—e-markets; H.3.5 [Online Information Services]: Web-based services—online reputation systems

General Terms
Experimentation, Management, Security

Keywords
e-markets, reputation systems, trust management, fraud evidence, fraud detection, logistic regression

1. INTRODUCTION
In the past few years, there has been a huge development of online commercial activity enabled by the Internet and World Wide Web (WWW). Electronic marketplaces, or just e-markets, such as Amazon\(^1\) and eBay\(^2\), have reached great popularity and revenue, emerging as very relevant model in the Business-to-Consumer (B2C) and Consumer-to-Consumer (C2C) e-commerce scenario. Amazon revenues reached US$ 19.17 billion in 2008, including a fast-growing income from selling Web Services to other companies. At eBay, sales reached US$15.7 billion in the second quarter of the year, with 84.5 million active users [5].

An e-market can be defined as a multi-party e-commerce platform intermediating buyers and sellers [13]. E-markets are therefore information systems intended to provide their users with online services that will facilitate information exchange and transactions. The development of online auction sites and other forms of electronic markets has created a new kind of online community, where people trade with

\(^1\)http://www.amazon.com  
\(^2\)http://www.ebay.com
each other in a potentially large scale. In this scenario, reputation plays an important role.

Reputation is the opinion (more technically, a social evaluation) of the public toward a person, a group of people, or an organization. Reputation can also be defined as the amount of trust inspired by a particular person or company in a specific domain of interest [20]. The widespread adoption of e-markets has highlighted several problems regarding trust and deception that must be addressed to keep these environments sound [6].

Major marketplace providers try to tackle the problem by introducing simple reputation mechanisms [30], which give an indication of how trustworthy a user is, based on his/her performance in previous transactions.

In this work, we focus on frauds against reputation systems. From a characteristic extraction based on expert knowledge [29], we use a set of characteristics for fraud detection in e-markets that was recently found [17]. We also describe a simple approach for identifying frauds by characteristics counting. Next, we enhance the characteristics set in this work and adopt a Logistic Regression Model that can deal with this set. We also apply a technique of optimization in the model generated by Logistic Regression. These approaches analyze characteristics of both the user and the negotiation processes that happen in the marketplace. We compare them by using actual data from a large Brazilian e-market and then checking the results with specialists to validate the outcomes. We found out that the performance of both methods is very promising, and they are useful for discovering a large number of fraudsters that have not been identified before.

The remainder of this article is organized as follows. Section 2 discusses related work. Section 3 briefly describes the TodasOferta marketplace, used in our study. In Section 4 we explain the characteristic extraction procedure for data used in this work. Section 5 shows the characteristic counting approach with some results. Section 6 presents a new approach that applies Logistic Regression with our case study and results, including Stepwise optimization. Finally, Section 7 shows our conclusions.

2. RELATED WORK

Electronic markets are getting more popular each day. One of the most common e-markets application is online auctions, which have been extensively studied lately. Several studies have focused on reputation systems and trust in online auctions. Some of them have analyzed the importance of reputation in auction outputs, mainly in final prices. Ba and Pavlou [2] investigate the effectiveness of reputation systems and how reputation correlates to auction results. They conclude that reputation plays an important role in trust and leads to higher ending prices.

Klos et. al [12] analyze the effect of trust and reputation over

3We use the term characteristic instead of feature, which is the term used in some references, because we consider the term characteristic more appropriated in statistical context

the profits obtained by intermediaries in electronic commercial connections. Different trust and distrust propagation schemes in e-commerce negotiations studied and evaluated by Guha et. al [8]. Resnick et al. [26] show that sellers with high reputation are more capable of selling their products, but the gains in final prices are reduced. Using a controlled experiment, Resnick et al. [27] study more accurately the impact of reputation on the auction outputs. The results show that, in general, bidders pay higher prices to sellers with higher reputation.

Several works investigate reputation systems and how they induce cooperative behavior in strategic settings. Dellarocas [3] has done a thorough review on this topic. While providing incentive to good behavior, reputation systems may also help eliciting deceptive behavior. In fact, some fraud-related studies rely on reputation information as a source of evidence of fraud [7].

This subject has long interested economists once sellers with good reputation can increase their prices because buyers pay for such reputation [11]. In the real world, reputation is built with time after some transactions, and sellers build a concept about themselves that becomes a reference to consumers. This historical record is used by future buyers when making a new transaction [25].

Reputation mechanisms are based on virtual opinions, given by people who generally do not know each other personally. Therefore electronic trust is more difficult to be established if compared to the real world. Taking a broad view, in these marketplaces a buyer’s reputation represents the probability of payment and a seller’s reputation represents the estimated probability of delivering the advertised item (product that has been bought) after the payment [11]. These probabilities are related to trust [21].

Resnick et al. [25] say that these reputation systems have three main problems: (i) buyers have little motivation to provide feedback to sellers; (ii) it is difficult to elicit negative feedback because it is common that people negotiate and solve problems before filling the evaluation in the system; (iii) it is difficult to assure honest reports. Since it is very easy to register in such systems, it is very easy to create a false identity that can be used to trade with other users and distort the reputation system.

As the feedback system is the basis of reputation in these marketplaces and gives information that is used before the moment the transaction happens, it is easy for fraudsters to make artificial transactions so that they can have a good reputation score. Basically, this artificial score can be used to deceive buyers who pay and do not receive the right product or it can be used to sell more goods because the seller will have favorable reputation [25]. Considering this situation, marketplaces should have tools to identify fraudsters, in order to protect honest users. Users who interact with fraudsters may have their reputation affected too [21]. Gavish and Tucci [6] show that buyers who are victims of frauds will decrease their volume of transactions, which it is not profitable to the marketplaces.
One of the pioneers on empirical and statistical approach on bankruptcy prediction, Ohlson [22] was probably the first academic researcher to apply Logistic Regression in the field. In his work, not only was he able to detect what was the most important evidence for bankruptcy prediction, but also find fraud an/or error evidence since he observed that “the reports of the misclassified bankrupt firms seem to lack any warning signals of impending bankruptcy." Since then, many other works have applied the method in fraud detection [14, 15, 16, 28]. In particular, Vlae et al. applied the method, in a pool of many other methods, for automobile insurance fraud data and concluded that “noteworthy is the good overall performance of the relatively simple and efficient techniques such as logit..." (Logistic Regression).

3. MARKETPLACE DESCRIPTION

This section describes TodaOferta\(^1\), which is a marketplace developed by the largest Latin America Internet Service Provider, named Universo Online (UOL)\(^2\). It also defines some basic concepts related to the marketplace. TodaOferta \([23]\) is a website for buying and selling products and services on the Web. Table 1 shows a short summary of the TodaOferta dataset. It embeds a significant sample of users, listings, and negotiations. Due to a confidentiality agreement, the quantitative information of this dataset can not be presented.

<table>
<thead>
<tr>
<th>Coverage (time)</th>
<th>Jun/2007 to Jul/2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>#categories (top-level)</td>
<td>32</td>
</tr>
<tr>
<td>#sub-categories</td>
<td>2,189</td>
</tr>
<tr>
<td>Average listings per user</td>
<td>4.63</td>
</tr>
<tr>
<td>Average listings per seller</td>
<td>42.48</td>
</tr>
<tr>
<td>Negotiation options</td>
<td>Fixed Price and Auction</td>
</tr>
</tbody>
</table>

Table 1: TodaOferta Dataset - Summary

Users correspond to buyers and sellers interested in making transactions in the marketplace. Listings are created by sellers to advertise products or services at a fixed-price or in an auction. When a buyer is interested in a listing he/she starts a negotiation. With a fixed-price listing, the negotiation automatically starts a transaction, indicating that buyer and seller should transact the good at the advertised price. With an auction, the winning bid will become a transaction when the auction finishes. Unlike eBay, where auctions represent almost 50% of all transactions [9], in TodaOferta auctions account for less than 2% of all transactions, since the vast majority of listings are fixed-price.

There are 32 top-level categories in TodaOferta, which include 2,189 sub-categories providing a variety of distinct products and services, from collectibles to electronic and vehicles. The current top sales sub-categories are cell phones, MP3 players and pen drives.

The TodaOferta marketplace employs a quite simple reputation mechanism. After each negotiation, buyers and sellers qualify each other with a rate of value 1 (positive), 0 (neutral), or -1 (negative). User’s reputation is defined as the sum of all qualifications received by him/her. Feedbacks from a same user are considered only once when computing the reputation score. Reputation systems are useful to communicate trust in electronic commerce applications. As TodaOferta does not charge sellers after transactions, it is quite simple to generate artificial transactions just to improve this punctuation. However, TodaOferta provides other information about sellers and buyers that can be used to identify trustful and distrustful users as well (e.g., time since the user is registered, comments left by users who negotiated with him/her).

Basically, sellers try to cheat the reputation system with two different purposes. The first one is to improve their sales and, consequently, their profits because his or her reputation in the marketplace seems better than it actually is. The second purpose is to exploit the good reputation that he or she has to commit other types of fraud, in general related to finance damage to buyers.

Another relevant feature of TodaOferta is the integration with a payment system which includes an escrow mechanism. This escrow system is named PagSeguro\(^4\) and it is very similar to PayPal\(^5\). If a seller uses the payment system, he can configure his listings so that he will receive payments through it. However, with the escrow feature, the seller will receive the payment only after the buyer has received the product or after 15 days. So, if a seller has enabled the payment system, he is more likely to be trustworthy, since he allows the buyer to block the money if the product was not delivered properly.

4. CHARACTERISTIC EXTRACTION

In this section we describe the procedures related to characteristic extraction that is the first part of the methodology described by Maranzato [19].

Given the importance of reputation systems, we decided to focus our experiments on identifying and evaluating characteristics that can be indicative of fraud in such systems. We present here a method for extracting fraud evidence from transactions and the reputation systems.

First we gathered a real dataset from TodaOferta (see Table 1) and a list of all users that were blocked for infringing the rules of the marketplace. Each item of this list contains a label describing the reason why the user was blocked. As our goal is to identify users that defraud the reputation system, we define a set \(FRS\) that contains the users blocked specifically for this reason and \(FRST\) the transaction with this fraud evidence. We also define a set \(AP\) containing all users blocked for any kind of fraud and \(AP\cap T\) the transaction. We consider the remaining users in the system as “not fraud” and put them in a set \(NF\) and \(NF\cap T\) the transaction.

\(^1\)http://www.todaoferta.com.br
\(^2\)http://www.uol.com.br
\(^4\)http://www.pagseguro.com.br
\(^5\)http://www.paypal.com
with no fraud evidence. Considering this, we can represent:

\[
FRS \subseteq AFr \\
AFr \cup NFr = \text{All Users.}
\]

One user is considered fraudulent if he/she participates in at least one fraudulent transaction, as a buyer or a seller. Thus, the transactions in which a given user is involved determine which set the user will be in. We organize the transactions in sets analogous to the ones defined for users. A user is in \(FRS\) if he/she is the buyer or the seller in at least one transaction in \(FRST\). If this is not the case, but the user is involved in at least one transaction in \(AFrT\), he/she is in \(AFr\). Users in \(NFr\) are those who only have transactions in \(NFrT\).

With this defined set, we start the activities related to gathering information to the process of characteristic extraction. Then we interviewed specialists in fraud detection in this marketplace to understand their procedures and to identify which evidence we should consider when looking for users that were trying to cheat the reputation system. Most of their work is about reaction to denunciations. The specialists listed a set of characteristics that they analyzed to identify fraudulent transactions but also pointed out that all these characteristics can also occur in honest transactions. During the interviews we suggested new characteristics that could be used in fraud detection and we included them in our tests – some of them were not useful. We also use our experience to try some characteristics. This kind of approach is defined by Duda and Hart [4] as prior knowledge, that is one of the sub-problems of pattern recognition. We also faced on some other sub-problems like feature extraction, overfitting and noise, for example.

It important to emphasize that we decided to consider only transactions with positive feedbacks from buyers, since the positive feedback is the main goal of frauds to the reputation system and that is the type of feedback which affects the punctuation of the feedback system. We plan to consider other types of feedback in a future work.

After analyzing the dataset, the mechanics of this marketplace and the information collected during the interviews, we considered five main events to be taken into account in a fraud detection process:

1. Seller’s registration;
2. Buyer’s registration;
3. Listing publication;
4. Transaction;
5. Feedback from Buyer to Seller\(^6\).

A timeline of these events can be seen in Figure 1. As you can see, we use not only the transaction information but all interactions between buyer and seller in the marketplace.

Another important concept is that in electronic marketplaces, transactions between users can be represented as a graph (see Figure 2), with a node for each user and an edge for one (or more) transactions between two users.

Using connection information available about the buyer and the seller and considering the events previously described we have found twelve characteristics the are indicative of frauds. We started from two connection attributes of each event: workstation identifier\(^7\) and IP address. Then we took the three events related to the buyer (Buyer’s registration, Transaction and Buyer’s feedback) and combined with the two events related to the seller (Seller’s registration and Listing publication), obtaining six combinations to be verified. We show these characteristics in Table 2, presenting an explanation of why each one can be considered a good characteristic of fraud and a warning about occurrences in legitimate transactions. As an example, characteristic SWLB is detected when we observe the same workstation identifier when the seller created the listing and when the buyer registered. Similar comparison is done for IP address in SILB.

We also extracted five other characteristics that can not be described by Boolean values like the ones presented in Table 2, which we list below:

- Quick Feedbacks from Buyers, in less than \(N\) hours after transaction (QFB);
- Small Rate of Visits per Transactions, smaller than \(N\) (SRVT);
- Short Interval for Transactions in the same Listing during \(N\) hours (SITL);
- Same domain in e-mails from buyers in the same listing considering \(N\) transactions (UDTB);
- E-mails with the same domain between sellers and buyers considering \(N\) transactions (SDBS);

In all the situations listed above, we can convert these characteristics into Boolean values by establishing a threshold for the value of \(N\) in each case.

These characteristics are named positive characteristics because they indicate evidence of fraud. On the other hand, we\(^7\) Due to confidentiality, we can not give more details about how this identifier is determined.

\(^6\)In this work, we are not considering feedbacks from sellers because they do not benefit sellers.

\(^7\)Due to confidentiality, we can not give more details about how this identifier is determined.
Table 2: List of Characteristics

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Suspection</th>
<th>Warning</th>
<th>Code</th>
<th>Situations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same workstation identifier</td>
<td>Seller and buyer used the same computer</td>
<td>They could have used a public computer</td>
<td>SWLB</td>
<td>Listing and Buyer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SWSB</td>
<td>Seller and Buyer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SWLT</td>
<td>Listing and Transaction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SWLF</td>
<td>Listing and Feedback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SWSF</td>
<td>Seller and Feedback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SWST</td>
<td>Seller and Transaction</td>
</tr>
<tr>
<td>Same IP Address</td>
<td>Seller and buyer used the same computer</td>
<td>They could have used a proxy or a public computer</td>
<td>SILB</td>
<td>Listing and Buyer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SISB</td>
<td>Seller and Buyer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SILT</td>
<td>Listing and Transaction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SILF</td>
<td>Listing and Feedback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SISF</td>
<td>Seller and Feedback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIST</td>
<td>Seller and Transaction</td>
</tr>
</tbody>
</table>

Table 2: List of Characteristics

found two characteristics that decrease the chance of fraud. We name them as *negative characteristics* and they are:

- Seller Recognition (REC1). This recognition can be done by editorial analyses from the marketplace, for example;
- Transaction paid through integrated escrow system (TCPS), that is *PagSeguro*;

All positive and negative characteristics are listed in Table 3.

Our next step is to expand this evidence from transactions and events to the sellers because the seller is the target of fraud detection. As we mentioned before, specialists consider that a user is fraudulent if he/she participates in at least one fraudulent transaction, as a seller or a buyer. With this approach that uses just one positive characteristic we reach 96.8% of sellers in FRST. Besides, we also reach 78.5% of users in $AFr - FRST$. Unfortunately, we also hit 54.3% of user that were not pointed as fraudsters (users in $NFrT$), which shows us that only one fraud characteristic (one characteristic among all positive seventeen we have obtained) is a weak information to give certainty about a fraud behavior.

Consider a characteristics and let $F$ be the set of all transactions that have this characteristic. We count how many transactions in $F$ are also in $FRST$ and in $NFrT$, and compute their respective probabilities:

$$p_1 = \frac{|F \cap FRST|}{|FRST|}$$
$$p_2 = \frac{|F \cap NFrT|}{|NFrT|}.$$

In order to evaluate the discriminating power of this characteristic, we compute the *odds ratio* between the classes $FRST$ and $NFrT$. The odds ratio is a measure that compares the probability of an event occurring in one group with the probability of it occurring in another group. If the probabilities of the event occurring in each of the groups are $p_1$ and $p_2$, then the odds ratio is:

$$\frac{p_1}{(1-p_1)} = \frac{p_1(1-p_2)}{p_2(1-p_1)}.$$

In this work, we only consider positive characteristics with odds ratio at least 3. For negative ones, we just invert the ratio and we use the same threshold. This simple approach
Characteristic | Description | Type
--- | --- | ---
QFB | Quick Feedbacks from Buyers, in less than \(N\) hours after transaction | Positive
REC1 | Seller Recognition | Negative
SDBS | E-mails with the same domain between sellers and buyers considering \(N\) transactions | Positive
SILB | Same IP Address – Listing and Buyer | Positive
SILF | Same IP Address – Listing and Feedback | Positive
SILT | Same IP Address – Listing and Transaction | Positive
SISB | Same IP Address – Seller and Buyer | Positive
SISF | Same IP Address – Seller and Feedback | Positive
SIST | Same IP Address – Seller and Transaction | Positive
SITL | Short Interval for Transactions in the same Listing during \(N\) hours | Positive
SRVT | Small Rate of Visits per Transactions, smaller than \(N\) | Positive
SWLB | Same workstation identifier – Listing and Buyer | Positive
SWLF | Same workstation identifier – Listing and Feedback | Positive
SWLT | Same workstation identifier – Listing and Transaction | Positive
SWSB | Same workstation identifier – Seller and Buyer | Positive
SWSF | Same workstation identifier – Seller and Feedback | Positive
SWST | Same workstation identifier – Seller and Transaction | Positive
TCPS | Transaction paid through integrated escrow system | Negative
UDTB | Same domain in e-mails from buyers in the same listing considering \(N\) transactions | Positive

Table 3: Complete list of Characteristics

is useful to to validate the characteristics and we introduce it in the next section, since it is based on counting the number of positive characteristics of a transaction.

5. CHARACTERISTICS COUNTING

This section explains a characteristic counting approach to validate characteristics in a fraud detection process. We can take the positive characteristics introduced in Section 4 and check if that the presence of only one of them is enough to determine that a transaction is fraudulent.

Now we can find out how a minimal number of characteristics \(k\) can be used as a strong evidence of fraud. First we rank the characteristics in decreasing order of their corresponding odds ratios. Iterating \(k\) up to the 17 characteristics, we compute the set \(K\) of sellers that participate in transactions with at least \(k\) characteristics. These characteristics are the positive ones listed in Table 3 and they are natural candidates for investigation.

Using this simply composed characteristic as a classification/ranking criteria, we apply the usual measures of precision, recall and F-measure, used for classifiers. The percentage of sellers in \(FRS\) that are in \(K\) is the recall. The percentage of sellers in \(K\) that are in \(FRS\) is the precision. The harmonic mean of recall and precision is the F-measure, which evaluates the usual trade off between precision and recall, providing a better measure to comparison. These results are in Table 4 and they will be compared to the Logistic Regression Model approach next in Section 6.

We saw that the best value of measure occurs when we reach 60.3% of sellers, with a precision of only 33.0%. In terms of a global metric for this ranking, we have reached an average

<table>
<thead>
<tr>
<th>(k)</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1.3%</td>
<td>80.0%</td>
<td>0.026</td>
</tr>
<tr>
<td>16</td>
<td>2.7%</td>
<td>66.7%</td>
<td>0.051</td>
</tr>
<tr>
<td>15</td>
<td>6.7%</td>
<td>66.7%</td>
<td>0.121</td>
</tr>
<tr>
<td>14</td>
<td>10.3%</td>
<td>59.6%</td>
<td>0.176</td>
</tr>
<tr>
<td>13</td>
<td>15.3%</td>
<td>53.5%</td>
<td>0.238</td>
</tr>
<tr>
<td>12</td>
<td>19.0%</td>
<td>49.1%</td>
<td>0.274</td>
</tr>
<tr>
<td>11</td>
<td>24.0%</td>
<td>49.0%</td>
<td>0.322</td>
</tr>
<tr>
<td>10</td>
<td>30.0%</td>
<td>45.9%</td>
<td>0.363</td>
</tr>
<tr>
<td>9</td>
<td>33.7%</td>
<td>42.6%</td>
<td>0.376</td>
</tr>
<tr>
<td>8</td>
<td>41.0%</td>
<td>38.7%</td>
<td>0.398</td>
</tr>
<tr>
<td>7</td>
<td>47.0%</td>
<td>35.2%</td>
<td>0.402</td>
</tr>
<tr>
<td>6</td>
<td>52.0%</td>
<td>34.0%</td>
<td>0.411</td>
</tr>
<tr>
<td>5</td>
<td>60.3%</td>
<td>33.0%</td>
<td>0.427</td>
</tr>
<tr>
<td>4</td>
<td>70.7%</td>
<td>28.4%</td>
<td>0.405</td>
</tr>
<tr>
<td>3</td>
<td>79.0%</td>
<td>22.3%</td>
<td>0.348</td>
</tr>
<tr>
<td>2</td>
<td>89.3%</td>
<td>18.0%</td>
<td>0.300</td>
</tr>
<tr>
<td>1</td>
<td>100.0%</td>
<td>12.4%</td>
<td>0.220</td>
</tr>
</tbody>
</table>

Table 4: Sellers with at least \(k\) characteristics
precision\textsuperscript{8} of 38.9%.

\section{LOGISTIC REGRESSION}

This section presents the Logistic Regression Model and its application. First we describe some concepts about it (Section 6.1) and then we describe our case study applying it to actual data from \textit{TodaOferta} to rank all sellers considering their estimated fraud probability (Section 6.2). It is important to say that we consider this problem of identifying fraudsters as a ranking problem instead of a classification problem that typically classifies the results in some predefined sets. Our intention is to generate a list that shows all sellers ordered by the percentage of one seller to be considered fraudster.

\subsection{Definition}

One of the greatest advantage of the Logistic Regression method over typical classification methods is that it provides a ranking ordering on the classified data, since it tries to predict the estimated fraud probability. Another important reason why we used this method and not others, is the fact that it is a quite natural extension to the odds ratio we have computed in the characteristics extraction analysis reported previously in Section 4. Applying to our data, the estimated probability of fraud ($p$) is represented by:

$$
p = \frac{1}{1 + e^{-z}}
$$

where, for constants $\beta_i$ and variables $x_i$, with $1 \leq i \leq 17$, we have:

$$z = \beta_0 + \beta_1 x_1 + \cdots + \beta_{17} x_{17}.$$

In fact, we have 17 variables (the fraud characteristics). And the constants $\beta_i$ are the best constants that fit the model to the data. The optimization procedures and details that are used for their obtainments are out of the scope of this work and we refer the reader to the work done by Hosmer and Lemeshow [10] for a better understanding of this method.

\subsection{Case Study and Results}

This section describes our case-study using a real dataset from \textit{TodaOferta} (described in Section 3) and it discusses the results of the experiments. In this section, we will apply part of the methodology proposed by Maranzato [19] to detect fraudsters against reputation systems in e-markets. We will compare and optimize some applications of the Logistic Regression Model.

It is important to say that the approach of sorting sellers by the number of positive characteristics of fraud has limitations if you compare it to the Logistic Regression Model that we apply in this work and another previous work [17].

The first limitation is that it is difficult to consider negative characteristics, that is, the percentage of the characteristic in \textit{NFrT} is significantly higher than in \textit{FRST} because it only sums each one without considering if its positive or negative evidence of fraud.

Another limitation is related to characteristics defined by thresholds. The counting characteristics method cannot deal with continuous values but only binary ones, because it considers only a binary response for a given characteristic of fraud. One example is the period between the transaction and the evaluation. We saw that the shorter the period evaluated is, the greater the chances of fraud. If we define a threshold we cannot use these variations of time as an input value to the analysis.

It is also possible to apply the Logistic Regression Model to consider percentages of occurrence of one characteristic in the transactions instead of just considering the characteristic of the seller. It is important to remember that it is a rule of the specialists in the investigation process in \textit{TodaOferta}, but considering as fraudster a seller that has only one suspicious transaction in many of them can generate a lot of false positives. Afterwards, we will see that this information is used to improve the ranking performance.

In order to apply Logistic Regression Model the first task is to prepare the dataset. To make a list of all sellers, for each one, we check if there is at least one transaction that contains the characteristics described in Section 4. We use 1 to indicate this existence and 0 otherwise. Afterwards, we improve this list with two non-fraud characteristics: an indication of if the seller passed for an editorial analysis\textsuperscript{9} and the percentage of transactions that are effectively done by \textit{PagSeguro}. To complete that list, we compute the percentage of transactions that has a characteristic for each seller.

The next step is to build the dataset that is going to be used as input to the Logistic Regression Model. We decided to adopt the full data for training and also for testing. The fact that the obtained performance numbers are not trustable because the generated model suffers from overfitting is not a problem anyway since the non-fraud annotations are not trustworthy, as we discovered in our previous work [17, 18]. We are aware of the problem, but we are interested in ranking sellers by estimated fraud probability and using the full data is better for this purpose. The real performance numbers of the ranking are those obtained by manual verification by specialists analysis.

It is important to say that both for training and testing datasets we excluded all transactions in \textit{AFrT} that are not in \textit{FRST}, since our focus is to discover fraudsters in reputation systems and these transactions can not be considered \textit{NFrT} neither \textit{FRST}. We believe that improvements in our work could handle \textit{AFrT} too, but here our focus is just \textit{FRST}. That is the reason to discard frauds that are not related to

\textsuperscript{9}\textit{TodaOferta} has a program in which sellers can send their documentation for analysis to get a certificate. It indicates that the seller really exists and follows the rules of marketplaces.
In order to apply the Logistic Regression Model to rank sellers, we use the free software for statistical computing R – for more details see http://www.r-project.org. Using command glm provided by this, we consider FRS as a dependent variable, considering the selected characteristics and using the same dataset for training and testing. Hence, we obtain the estimated probability for a seller to be in FRS, and we sort the sellers list from highest to lowest estimated fraud probabilities. It is important to remember that in this first application we have just used the same 17 characteristics that can be used in Characteristic Counting described in Section 5.

With this sorted list of sellers, we compute precision, recall and F-measure for each percentage of the list. One short summary of these results is listed in Table 5. We saw that checking between 8% and 10% of the sellers sorted by the estimated fraud probability, we can achieve the best value of F-measure, covering around 50% of fraudsters against reputation system in this marketplace. We will name this first application of Logistic Regression Model as “APP1” for further comparison.

<table>
<thead>
<tr>
<th>% of sellers</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>10.7%</td>
<td>82.1%</td>
<td>0.189</td>
</tr>
<tr>
<td>2%</td>
<td>17.7%</td>
<td>67.9%</td>
<td>0.280</td>
</tr>
<tr>
<td>3%</td>
<td>23.3%</td>
<td>59.8%</td>
<td>0.336</td>
</tr>
<tr>
<td>4%</td>
<td>29.3%</td>
<td>56.4%</td>
<td>0.386</td>
</tr>
<tr>
<td>5%</td>
<td>33.3%</td>
<td>51.3%</td>
<td>0.404</td>
</tr>
<tr>
<td>6%</td>
<td>38.0%</td>
<td>48.7%</td>
<td>0.427</td>
</tr>
<tr>
<td>7%</td>
<td>41.0%</td>
<td>45.1%</td>
<td>0.429</td>
</tr>
<tr>
<td>8%</td>
<td>46.7%</td>
<td>44.9%</td>
<td>0.458</td>
</tr>
<tr>
<td>9%</td>
<td>49.7%</td>
<td>42.5%</td>
<td>0.458</td>
</tr>
<tr>
<td>10%</td>
<td>52.7%</td>
<td>40.5%</td>
<td>0.458</td>
</tr>
<tr>
<td>15%</td>
<td>65.0%</td>
<td>33.3%</td>
<td>0.441</td>
</tr>
<tr>
<td>20%</td>
<td>76.0%</td>
<td>29.2%</td>
<td>0.422</td>
</tr>
<tr>
<td>30%</td>
<td>83.7%</td>
<td>21.5%</td>
<td>0.341</td>
</tr>
<tr>
<td>40%</td>
<td>93.7%</td>
<td>18.0%</td>
<td>0.302</td>
</tr>
<tr>
<td>50%</td>
<td>97.7%</td>
<td>14.7%</td>
<td>0.256</td>
</tr>
<tr>
<td>60%</td>
<td>98.7%</td>
<td>12.4%</td>
<td>0.221</td>
</tr>
<tr>
<td>70%</td>
<td>98.7%</td>
<td>10.7%</td>
<td>0.193</td>
</tr>
<tr>
<td>80%</td>
<td>98.7%</td>
<td>9.4%</td>
<td>0.171</td>
</tr>
<tr>
<td>90%</td>
<td>98.7%</td>
<td>8.3%</td>
<td>0.154</td>
</tr>
<tr>
<td>100%</td>
<td>100.0%</td>
<td>7.6%</td>
<td>0.141</td>
</tr>
</tbody>
</table>

Table 5: APP1 - Sellers ordered by probability of fraud (Recall, Precision and F-measure)

Analyzing the results of APP1, we can see that the average precision is 45.5% - it represents an increment of 16.1% if you compare to the Characteristic Counting approach (CC).

Next, we are going to compare the results of the Characteristic Counting approach (see Section 5) with Logistic Regression Model in three different applications: the first one uses the same 17 characteristics from Section 4 (which we have already done previously) and the second one uses percentages, negative characteristics and characteristics with continuous distribution, that represents 37 different variables (APP2) and the third one optimizes the second model using Stepwise Regression (APP3).

For a given data, this optimization basically selects the variables upon which the best model is based. In this work, we use backward regression, which involves starting with all candidate variables and testing them one by one for statistical significance, deleting any that are not significant to the model. In this case, the selection criteria is based on Akaike’s Information Criterion (AIC)\(^\text{10}\). We refer the reader to the work from Hosmer and Lemeshow [10] for a better understanding of this optimization method.

In those three applications of Logistic Regression, we compute the F-measure of both for each portion of sellers, in the same way that we did in the first application (APP1). We also compute average precision for the entire rank to have a single metric for each situation.

In Figure 3 we can also see that when using more variables (APP2 and APP3), the precision is better if you compare with the first one that was using only positive and binary characteristics (APP1). In Table 6 we have the values of average precision in this 4 different ways of ranking – one with Characteristic Counting and 3 with the estimated probability provided by the application of Logistic Regression. We see that when we apply a Stepwise Regression the precision values decrease a little, which shows that this kind of optimization, with the original annotation in this dataset, does not affect the global performance of this ranking.

![](image1.png)

APP3 resulted in 17 variables and they are different from APP1. With these results, that are in theory the best values that we can achieve with the characteristics that we have extracted in this dataset, we return to specialists to ask them to check what the actual performance of the output of these

\(^{10}\)For more details about Akaike’s information criterion, please check the reference from Akaike [1]. The command in software R that performs this model selection is stepAIC.
applications of Logistic Regression Model is, especially at the top of the list. In our previous work we discovered more fraudsters than the original annotation [17, 18] so we know in advance that the performance metrics could be better. It is important to say that the best revision process would be if it were possible to use the list generated by APP3, but specialists also checked previous lists sorted by characteristics counting when we were validating those characteristics. After that revision, we reach an average precision of 78.0% (we had around 53% before revision in APP2 and APP3).

As the number of elements in the predefined sets has changed because new fraudsters were detected, we decided to build a new model using the same variables of the second approach that considers 37 characteristics, but using the values of the reference variable after specialists’ revision (FRS Reviewed).

After that, we decided to optimize the model by applying Stepwise Regression to find the best model that fits in this dataset with the new fraudsters set. This optimization generates a model with 25 out of 37 variables and its performance is shown in Table 7. In the final model, only one characteristic extracted was not used (SILF). We can see that either the binary characteristic, that was described in Section 4, or its respective percentage in the transactions is present. It show that the process of characteristic extraction was correct. These variables are:

\[
x_1 = \text{pctSITL}, \ x_2 = \text{SWLB}, \ x_3 = \text{pctSWLB}, \ x_4 = \text{SWLT}, \ x_5 = \text{pctSWLF},
\]
\[
x_6 = \text{SWSB}, \ x_7 = \text{SWST}, \ x_8 = \text{SWSF}, \ x_9 = \text{SDBS}, \ x_{10} = \text{pctSDBS},
\]
\[
x_{11} = \text{pctSILB}, \ x_{12} = \text{SILT}, \ x_{13} = \text{SILT}, \ x_{14} = \text{pctSILB}, \ x_{15} = \text{SFT}
\]
\[
x_{16} = \text{pctSFT}, \ x_{17} = \text{SIF}, \ x_{18} = \text{QBF}, \ x_{19} = \text{pctQBF}, \ x_{20} = \text{UDTB},
\]
\[
x_{21} = \text{pctUDTB}, \ x_{22} = \text{SRVT}, \ x_{23} = \text{TCP}, \ x_{24} = \text{SITL}, \ x_{25} = \text{REC1}
\]

Analyzing the results in Table 7 we see that the precision of this model is 100% when we consider 9% of the sellers in the sorted list. It is also possible to see that the best value of F-measure is when we check 14% of the sellers. At this

<table>
<thead>
<tr>
<th>% of sellers</th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>6.1%</td>
<td>100.0%</td>
<td>0.115</td>
</tr>
<tr>
<td>2%</td>
<td>12.2%</td>
<td>100.0%</td>
<td>0.218</td>
</tr>
<tr>
<td>3%</td>
<td>18.4%</td>
<td>100.0%</td>
<td>0.310</td>
</tr>
<tr>
<td>4%</td>
<td>24.5%</td>
<td>100.0%</td>
<td>0.393</td>
</tr>
<tr>
<td>5%</td>
<td>30.6%</td>
<td>100.0%</td>
<td>0.469</td>
</tr>
<tr>
<td>6%</td>
<td>36.7%</td>
<td>100.0%</td>
<td>0.537</td>
</tr>
<tr>
<td>7%</td>
<td>42.9%</td>
<td>100.0%</td>
<td>0.600</td>
</tr>
<tr>
<td>8%</td>
<td>49.0%</td>
<td>100.0%</td>
<td>0.658</td>
</tr>
<tr>
<td>9%</td>
<td>55.1%</td>
<td>100.0%</td>
<td>0.711</td>
</tr>
<tr>
<td>10%</td>
<td>60.9%</td>
<td>99.3%</td>
<td>0.756</td>
</tr>
<tr>
<td>11%</td>
<td>66.9%</td>
<td>99.3%</td>
<td>0.799</td>
</tr>
<tr>
<td>12%</td>
<td>72.7%</td>
<td>98.9%</td>
<td>0.838</td>
</tr>
<tr>
<td>13%</td>
<td>76.9%</td>
<td>96.6%</td>
<td>0.857</td>
</tr>
<tr>
<td>14%</td>
<td>80.4%</td>
<td>93.8%</td>
<td>0.866</td>
</tr>
<tr>
<td>15%</td>
<td>82.9%</td>
<td>90.3%</td>
<td>0.864</td>
</tr>
<tr>
<td>20%</td>
<td>89.0%</td>
<td>72.7%</td>
<td>0.800</td>
</tr>
<tr>
<td>30%</td>
<td>94.5%</td>
<td>51.5%</td>
<td>0.666</td>
</tr>
<tr>
<td>40%</td>
<td>98.0%</td>
<td>40.0%</td>
<td>0.568</td>
</tr>
<tr>
<td>50%</td>
<td>99.1%</td>
<td>31.7%</td>
<td>0.481</td>
</tr>
<tr>
<td>60%</td>
<td>100.0%</td>
<td>26.5%</td>
<td>0.422</td>
</tr>
<tr>
<td>70%</td>
<td>100.0%</td>
<td>23.0%</td>
<td>0.374</td>
</tr>
<tr>
<td>80%</td>
<td>100.0%</td>
<td>20.2%</td>
<td>0.336</td>
</tr>
<tr>
<td>90%</td>
<td>100.0%</td>
<td>17.9%</td>
<td>0.304</td>
</tr>
<tr>
<td>100%</td>
<td>100.0%</td>
<td>16.1%</td>
<td>0.278</td>
</tr>
</tbody>
</table>

Table 7: APP4 - Sellers ordered by probability of fraud (Recall, Precision and F-measure)

Figure 3: Comparison between Characteristic Counting approach and Logistic Regression Model in different applications
point, we reach more than 80% of fraudsters with 93.8% of precision.

We have seen in Figure 3 that we are constantly improving precision values, especially on top of sellers’ list. But after specialists’ revision, this performance is much better, as we can see in Figure 4. This gain can also be verified in Table 6, where we observe the average precision values of each application. And in the last application (APP4), the average precision is 93%.

Figure 4: Comparison between Logistic Regression Model before and after specialists’ review

Another important tool to compare the performance of these different ways to apply Logistic Regression Model and the optimization done through the experiments in our dataset is to analyze precision in terms of recall.

Figure 5: Precision x Recall

In Figure 5 we see that using a model optimized to the annotation after specialists’ revision, the precision is better if we compare to the original annotation. It confirms that our procedures improve the fraud detection process because we generate a list of fraudsters sorted by an estimated probability that has high precision on the top of that list. In this scenario, it does not generate many false-positives and reduces the cost of fraud investigation.

Another benefit of using Logistic Regression Model is that it is possible to input the model with the dataset of reviewed analysis and rebuild it. Furthermore, it is possible to optimize the model with the new training dataset. We believe that it can improve the accuracy of the fraudsters’ list and be adapted to new transactions.

The results also show an increment of the average precision in the reviewed list compared to the original precision in 75% (from 53% to 93%). This indicates that our approach of optimization improves significantly the fraud detection process.

Considering the problem of overfitting that we have explained before in the beginning of this section, we are conducting some experiments splitting the dataset in training and testing data. We are sorting the transactions by a chronological date and using the first ones (or the older ones, in other words) to train and the last ones to test, simulating the actual scenario of the marketplace, where new transactions are occurring and these ones need to be investigated. The results that we are obtaining show that the precision remains in the same level that we have in this work – higher than 90% in average precision.

One important benefit of using Logistic Regression and optimization is that we can determine a threshold of F-measure (or precision or recall), to automatically disable some sellers. While specialists are checking our list, we observe that for the top ranked sellers, they just disable the fraudster, but as they continue checking, in some cases, they prefer to warn them, considering that seller defrauds the reputation system, but he has some transactions that are licit (this threshold determination can be subject of future work). This situation frequently happens when the seller has negative characteristics of fraud. It offers an opportunity to punish those sellers, to register fraud, and to give them another chance. It can be recorded and used in future events. It also allows them to reconsider some editorial recognition and the relationship that was established.

7. CONCLUSIONS

E-markets constitute an important research scenario due to their popularity and revenues over the last years. In this scenario, reputation plays an important role, mainly for protecting buyers from fraudulent sellers. A reputation mechanism tries to provide an indication of how trustworthy a user is, based on his/her performance in previous transactions.

In online marketplaces, reputation is based on feedback systems that use the past transactions as reference to show user performance with the intention of providing more information for future transactions. Mostly, fraud detection is done through reactive procedures where fraud experts conduct an investigation from a user claim. This work is focused on support decision for fraud detection against the reputation systems as a complement to fraud experts decisions.

Here, we apply such a set of characteristics for fraud detection to the e-market reputation systems. Besides, we describe and evaluate two approaches for identifying frauds, one based on fraud characteristics counting and another one
based on Logistic Regression modeling. Moreover, we enhance the characteristics set with non-fraud characteristics that Logistic Regression can deal with but Characteristics Counting cannot. We compare both approaches using actual data from a large Brazilian e-market (TodaOferta).

We apply Logistic Regression Model to these set of characteristics. The method naturally ranks all sellers considering their fraud against the reputation system estimated probability. We also see that it is possible to optimize the model using stepwise regression to improve the quality of the ranking. The output of this application is a list of sellers sorted by the estimated fraud probability.

This list allows experts to prevent fraud instead of just reacting. In the end, using the rank provided by Logistic Regression, we increased by 112% the number of identified fraudsters in TodaOferta marketplace. And when we optimize the model we reach an average precision of 93%.

As future work, we want to apply the same methodology to identify other types of fraud along with the ones in reputation systems. In particular, we are interested in finding correlation between frauds in reputation systems and other types of frauds. The idea of using network-based metrics [24] to complement the current characteristics of fraud seems also to be promising. We are planning a deeper analysis of the final obtained logistic model, checking which characteristics are stronger than others and analyzing their coefficients $\beta$ [10], before trying other classification and/or ranking methods.

8. ACKNOWLEDGMENTS
This work was partially sponsored by Universo Online S.A. - UOL (http://www.uol.com.br) and partially supported by the Brazilian National Institute of Science and Technology for the Web (CNPq grant no. 573871/2008-6), CAPES, CNPq, Finep, and Fapemig. We also thank Aline Pereira and Rodnei Lozano, from UOL, for their support on the analysis and validation of our results.

9. REFERENCES


About the authors:

Rafael Plana Maranzato is a project manager in the department of research and development of ecommerce products at Universo Online (UOL), the largest Brazilian Internet portal. He holds a bachelor degree in mathematics at Fundação Santo André-SP (FSA) in 1999 and received his master degree in 2010 at the Mathematics & Statistics Institute (IME) of University of São Paulo (USP) - his research was about fraud detection in reputation systems in marketplaces. He is also interested in agile development and project management.

Adriano C. Machado Pereira is an Adjunct Professor at Federal Center of Technological Education of Minas Gerais (CEFET-MG), Brazil. He received his bachelor degree in Computer Science at UFMG in 2000, his MSc. in 2002, and his Ph.D. in 2007. His research interests include e-Business, Information and System’s Credibility, Distributed Systems, Web 2.0, Social Networks, Performance of Computer Systems, Web Technologies, Workload Characterization, Formal Methods, and Business Intelligence. He has worked as a consultant for the Brazilian government and United Nations Development Programme (UNDP) in a project of an Open Source Cooperation Network for e-governance for Latin America. He is also a member of the Brazilian National Institute of Science and Technology for the Web (www.inweb.org.br). He is also a Post-Doc researcher in Computer Science in a cooperated project with UOL, the largest Latin American Internet Service Provider.

Marden Neubert holds a bachelor degree and a Master's degree in Computer Science from Federal University of Minas Gerais (UFMG), Brazil. He has researched topics in Information Retrieval, Software Engineering and Fraud Detection. Currently he works as Research and Development Director at Universo Online (UOL), the largest Brazilian Internet portal.

A. Pereira do Lago, an assistant professor in the Computer Science Department of University of São Paulo, is one of Imre Simon's students, a prized Hungarian mathematician who helped to found Computer Science in Brazil. Imre improved the mathematical rigor that was already present in the student since he also obtained the third premium in the International Olympiads on Mathematics, in 1983. They both helped to solve a 20 years old conjecture in Automata Theory and Formal Languages, and their work was cited by important researchers like Mark Sapir and Nachum Dershowitz. Supervising other students on their Masters and/or PhD work, the former student has also been able to develop a quite rigorous research is areas as different as Operational Systems, Computational Biology, Advanced Data Structures, Information Retrieval, Formal Concept Analysis and Fraud Detection.
A tool for rapid development of WS-BPEL applications

Luca Cesari  
Dipartimento di Sistemi e Informatica  
Università degli Studi di Firenze  
cesari.luca@gmail.com

Rosario Pugliese  
Dipartimento di Sistemi e Informatica  
Università degli Studi di Firenze  
rosario.pugliese@unifi.it

Francesco Tiezzi  
Dipartimento di Sistemi e Informatica  
Università degli Studi di Firenze  
tiezzi@dsi.unifi.it

ABSTRACT
WS-BPEL is imposing itself as a standard for orchestration of web services. However, there are still some well-known difficulties that make programming in WS-BPEL a tricky task. In this paper, we present BîteC, a software tool we have developed for supporting a rapid and easy development of WS-BPEL applications. BîteC translates service orchestrations written in Bîte, a formal language inspired to but simpler than WS-BPEL, into readily executable WS-BPEL programs. We illustrate our approach by means of a few practical programming examples.

Categories and Subject Descriptors
D.2.2 [Software engineering]: Design Tools and Techniques—Computer-aided software engineering; D.3.1 [Programming Languages]: Formal Definitions and Theory—Syntax Semantics; D.3.4 [Programming Languages]: Processors—Compilers Parsing

Keywords
Service-Oriented Computing, Web services, Compilers

1. INTRODUCTION
In recent years, there has been an ever increasing acceptance of WS-BPEL [30] as a standard language for orchestration of web services, one of the most successful and well-developed implementations of the Service-Oriented Computing (SOC) paradigm. However, designing and developing WS-BPEL applications is a difficult and error-prone task. The language has an XML syntax which makes awkward writing WS-BPEL code by using standard editors. Therefore, many companies (among which e.g. Oracle and Active Endpoints) have equipped their WS-BPEL engines with graphical designers. Such tools are certainly suitable to develop simple business processes, but turn out to be cumbersome and ineffective when programming more complex applications. Further difficulties derive from the fact that WS-BPEL is equipped with such intricate features as concurrency, multiple service instances, message correlation, long-running business transactions, termination and compensation handlers. Most of all, WS-BPEL comes without a formal semantics and its specification document, written in ‘natural’ language, contains a fair number of acknowledged ambiguous features that may give rise to different interpretations. These ambiguities have led to engines implementing different semantics (see [25]) and, hence, have undermined portability of WS-BPEL programs across different platforms. Therefore, many research efforts have been devoted to provide WS-BPEL with a formal semantic (see, e.g., [27, 18, 33, 31, 24, 26, 20, 21]), although most of them do not deal with the complete language. Finally, the deployment procedure of WS-BPEL programs is not standardised, which further compromises portability. In fact, to execute a WS-BPEL program, besides the associated WSDL [17] document that describes the program’s public interfaces, different engines require different (and not integrable) process deployment descriptors, i.e. sets of configuration files that describe how the program should be deployed.

To overcome these difficulties, we have developed BîteC, a software tool that accepts as an input a specification written in the lightweight orchestration language Bîte [25] and returns the corresponding WS-BPEL program together with the associated WSDL and deployment descriptor files.

Bîte is closely inspired to WS-BPEL. It is the result of a tension between handiness and expressiveness. While the set of WS-BPEL constructs is not intended to be a minimal one, the design of Bîte, to keep the language manageable, only retains the core features of WS-BPEL. It follows that Bîte is simpler and more compact than WS-BPEL, although it maintains the same descriptive power. Using Bîte for initially specifying a service orchestration offers some significant advantages. From the one hand, Bîte textual notation is certainly more manageable than those, possibly graphical, notations proposed for WS-BPEL. From the other hand, Bîte is equipped with a formal operational semantics that clarifies all ambiguous and intricate aspects of WS-BPEL. Of course, to preserve such semantics on different WS-BPEL engines, the translation of Bîte programs into WS-BPEL ones has to be properly targeted to each specific engine.

BîteC further simplifies the programmers work by automating the deployment procedure. In fact, the returned files are properly packaged to be immediately executable in a WS-BPEL engine. Currently, these packages are intended
The handlers within a `<scope>` can be of four different kinds: `<faultHandler>`, to provide the activities in response to faults occurring during execution of the primary activity; `<compensationHandler>`, to provide the activities to compensate the successfully executed primary activity; `<terminationHandler>`, to control the forced termination of the primary activity; and `<eventHandler>`, to process message or timeout events occurring during execution of the primary activity. If a fault occurs during execution of a primary activity, the control is transferred to the corresponding fault handler and all currently running activities inside the scope are interrupted immediately without involving any fault/compensation handling behaviour. If another fault occurs during a fault/compensation handling, then it is re-thrown, possibly, to the immediately enclosing scope. Compensation handlers attempt to reverse the effects of previously successfully completed primary activities (scopes) and have been introduced to support Long-Running (Business) Transactions (LRTs). Compensation can only be invoked from within fault or compensation handlers starting the compensation either of a specific inner (completed) scope, or of all inner completed scopes in the reverse order of completion. The latter alternative is also called the default compensation behaviour. Invoking a compensation handler that is unavailable is equivalent to perform an empty activity.

A WS-BPEL program, also called (business) process, is a `<process>`, that is a sort of `<scope>` without compensation and termination handlers.

WS-BPEL uses the basic notion of partner link to directly model peer-to-peer relationships between services. This relationship is expressed at the WSDL level by specifying the roles played by each of the services in the interaction. However, the information provided by partner links is not enough to deliver messages to a business process. Indeed, since multiple instances of a same service can be simultaneously active because service operations can be independently invoked by several clients, messages need to be delivered not only to the correct partner, but also to the correct instance of the service that the partner provides. To achieve this, WS-BPEL relies on the business data exchanged rather than on specific mechanisms, such as WS-Addressing [22] or low-level methods based on SOAP headers. In fact, WS-BPEL exploits correlation sets, namely sets of correlation variables (called properties in WS-BPEL jargon), to declare the parts of a message that can be used to identify an instance. This way, a message can be delivered to the correct instance on the basis of the values associated to the correlation variables, independently of any routing mechanism.

We end this section by showing an auction service described in the official specification of WS-BPEL [30, Sect. 15.4]. This example will allow us to illustrate most of the language features, including correlation sets, shared variables, control flow structures, asynchronous communication and multiple start activities, and, through its implementation in Blite presented in Section 5.3, will permit a rough comparison between the two languages.

The auction service collects information from a seller and a buyer about a concluded auction, reports the auction result to an auction registration service, and then communicates...
the registration result to the seller and the buyer. The auction house process may be instantiated either by receiving the seller information or by receiving the buyer information. Indeed, the process is able of receiving the seller and buyer requests in a statically unpredictable order and in such a way that the first incoming message triggers the creation of a process instance which the subsequent request is delivered to. This requires the two starting receive activities to share a correlation set, which will be initialized with an auction identifier that the seller and the buyer have to provide when sending their requests. The auction house process passes the auction identifier to the auction registration service that, in its turn, returns the identifier in its answer to locate the proper process instance.

We report below the corresponding WS-BPEL program, where to make the reading of the code easier, we have omitted irrelevant details and highlighted the basic activities receive, invoke and assign.

```bpe1
<process name="auctionService" ...
<partnerLinks> ...
<variables> ...
<correlationSets>
  <correlationSet name="auctionIdentification"
               properties="as:auctionId" />
</correlationSets>
<sequence>
  <receive name="acceptSellerInformation"
    partnerLink="seller"
    portType="as:sellerPT"
    operation="submit"
    variable="sellerData"
    createInstance="yes">
    <correlations>
      <correlation set="auctionIdentification"
                   initiate="join" />
    </correlations>
  </receive>
  <receive name="acceptBuyerInformation"
    partnerLink="buyer"
    portType="as:buyerPT"
    operation="submit"
    variable="buyerData"
    createInstance="yes">
    <correlations>
      <correlation set="auctionIdentification"
                   initiate="join" />
    </correlations>
  </receive>
  <assign>
    <copy>
      <from>..
      ...
      ... http://example.com/auction/RegistrationService ...
      </from>
    </copy>
    <to partnerLink="auctionRegistrationService" />
  </assign>
  <copy>
    <from partnerLink="auctionRegistrationService"
      endpointReference="myRole" />
    <to>$auctionData.auctionHouseEndpointReference</to>
  </copy>
  <copy>
    <from>$sellerData.auctionId</from>
    <to>$auctionData.auctionId</to>
  </copy>
  <copy>
    <from>1</from>
    <to>$auctionData.amount</to>
  </copy>
  <invoke name="registerAuctionResults"
    partnerLink="auctionRegistrationService"
    portType="as:auctionRegistrationPT"
    operation="process"
    inputVariable="auctionData" />
  <invoke name="acceptBuyerInformation"
    partnerLink="buyer"
    portType="as:buyerAnswerPT"
    operation="answer"
    inputVariable="buyerAnswerData" />
</sequence>
</flow>
</sequence>
</process>
```

Notice that the buyer and the seller provide their endpoint references for the auction house process to respond properly in an asynchronous way. For similar reasons, the auction house process provides its own endpoint reference to the auction registration service.

### 3. Programming Services in Blite

A Blite program accepted by BliteC is composed of a Blite specification and a declarative part. The former focusses on the behavioural aspects of the orchestration, while the latter provides the implementation details (e.g. types, addresses, bindings, . . . ) that are necessary to deploy and execute the corresponding WS-BPEL program.

#### 3.1 Blite specification

Blite [25] is a prototypical orchestration language whose design is closely inspired to WS-BPEL. To keep the language manageable, the design of Blite only retains the core features of WS-BPEL. In fact, some aspects have been intentionally left out, including timeouts, synchronization dependencies within flow activities, event and termination handlers. Moreover, Blite only provides a simplified form of fault and compensation handling and only supports unnamed faults and the default compensation mechanisms.

Blite provides a formal description of service deployments by only keeping relevant implementation details. Thus, the roles played by service partners in a service interaction are explicitly indicated by partner links and partners, while such aspects as physical service binding (necessary to generate
the associated WSDL documents and deployment descriptors) are abstracted away and dealt with separately in the declarative part.

The syntax of Bitel accepted by BitelC is given in Figure 2. Services are structured activities built from basic activities, i.e., service invocation, service request processing, assignment, empty activity, fault generation and instance forced termination, by exploiting operators for conditional choice, iteration, sequential composition, parallel composition, pick and scope. A scope activity groups a primary activity $A$ together with a fault handling activity $A_f$ and a compensation activity $A_c$. Start activities are structured activities that initially can only execute receive activities. Sequence has higher priority (i.e. bind more tightly) than parallel composition and pick. Moreover, fault and compensation activities may be omitted from a scope construct, in which case they are intended to be three and empty, respectively.

Notation $\langle \cdot \rangle$ stands for tuples of objects, e.g. $\langle x_1, . . . , x_n \rangle$ denotes a tuple of variables (variables in the same tuple must be pairwise distinct). Partner links $pl$ can be either of the form $\langle \text{partner} \rangle$ or of the form $\langle \text{partner}_1, \text{partner}_2 \rangle$. Indeed, in one-way interactions a partner link indicates a single partner because one of the parties provides all the invoked operations. Instead, in asynchronous request-response interactions, partner links indicate two partners because the requesting partner must provide a callback operation used by the receiving partner to send notifications. Service partners used for receiving messages must be known at design-time, while the partners used to send messages in reply may be dynamically determined.

Besides asynchronous invocation, WS-BPEL also provides a construct for synchronous invocation of remote services. This construct forces the invoker to wait for an answer by the invoked service, that indeed performs a pair of activities receive-reply. In Bitel, this behaviour is rendered in terms of a pair of activities invoke-receive over the same operation executed by the invoker and a corresponding pair of activities receive-invoke executed by the invoked service.

Data can be shared among different activities through shared variables (ranged over by $x$, $x_1$, . . . ). The manipulable values are boolean, integer numbers (ranged over by int), strings (as usual, written within double inverted commas), partner links, and literals (defined in the declarative part and denoted by putting the symbol $\$ in front of the corresponding identifier). Expressions combine values and variables by means of boolean, arithmetic, comparison and string operators. Operators $\text{set}(x, "path")$ and $\text{get}(x, "path")$ can be used respectively in the left and right hand sides of an assignment to act on a specific element (indicated by path) of the XML message stored in the variable $x$. Both operators turn out to be quite useful for easily interacting with non-Bitel services (see Section 5.5).

Bitel specifications are finite compositions of definitions (that assign names to Bitel terms), containing at most one deployment definition. A deployment associates a correlation set, namely a (possibly empty) set of correlation variables, to a service. A service provides a ‘top-level’ scope (i.e. a scope that cannot be compensated) and offers a choice of alternative receives among multiple start activities.

We refer the interested reader to [25] for a formal account of the Bitel operational semantics.

### 3.2 Declarative part

The declarative part of a Bitel program specifies configuration data necessary to properly translate the Bitel specification into an executable WS-BPEL program. Notably, BitelC requires the user to insert only the strictly necessary data. The declarations must be included within $<$def and $>$, and can occur in any position within a Bitel program.

A declarative part has the following form:

```xml
<bitel
  ADDRESSES {
    myname => "base_for_namespaces*;
    myaddress => "base_for_service_url*;
  }
  IMPORTS {
    associations prefix => "url*;
  }
  VARIABLES {
    variable and message declarations
  }
  LITERALS {
    variable and message declarations
  }
  PARTNERLINKS {
    partner link type declarations
  }
}
```

where blocks ADDRESSES and VARIABLES are mandatory, while the other ones can be omitted.

Within the ADDRESSES block the user has to specify the base for the namespaces used inside the generated files (after the keyword myname) and the base for the address where the new service will be hosted (after the keyword myaddress).

To define a service orchestration it is often necessary to import data (e.g. type declarations) from documents (e.g. WSDL files) associated to other services. To this aim, the user can specify the addresses of the documents to be imported within the IMPORTS block, by associating to each imported document a namespace prefix that will be used in the subsequent declarations to refer to it. Notably, definitions belonging to standard namespaces (e.g. http://www.w3.org/2001/XMLSchema) are automatically imported and, hence, do not require any declaration.

Bitel variables are untyped, while WS-BPEL ones must be typed. Therefore, to enable an automated translation, the user has to declare the type of the variables (both local variables and messages) within the VARIABLES block. Local variables, that can be used to temporarily store data and manipulate them, are declared by associations of the form $x => \text{XML_Schema}_{\text{type}}$ (e.g. $x\text{\_city} => \text{xsd:string}$). Messages, i.e. tuples of variables used as either sending source or receiving target, can be declared in two ways:

- by using an imported message type. For example, in $\langle \text{x\_auctionId}, \text{x\_registrationId} \Rightarrow \text{reg:regResp}$, the message composed of variables $x\_auctionId$ and $x\_registrationId$ is typed as reg:regResp, that is defined in the (WSDL) document identified by the namespace prefix reg (defined in the IMPORTS block);
by generating a new message type. For example, in

```xml
<x_auctionId,x_creditCardNumber,x_phoneNumber> => gen:buyerReq, <id,cCNum,phone>,<xsd:int,xsd:string,xsd:string>;
```

message `x_auctionId,x_creditCardNumber,x_phoneNumber` is typed as `buyerReq`, that defines messages composed of one integer and two string parts, id, cCNum and phone, respectively. The namespace prefix `gen` indicates that the type must be generated. If the type of a message part is an element type defined in an XML schema not generated by `BliteC`, the keyword (El) must precede the type. For example, in

```xml
<num, scale, reqWeight> => gen:req,
<x:int, scale, request>,<xsd:integer,xsd:string, (El)num:GetCityWeatherByZIP>;
```

the element type `GetCityWeatherByZIP` is defined in the block types of the WSDL document identified by `get`. In a WS-BPEL program, literals (i.e. constant values) can be directly assigned to variables. Instead, in a `Blite` program, the sake of readability, literals must first be declared within the LITERALS block, as e.g.

```xml
litConv => [[<tem:FahrenheitToCelsius xmlns:tem="http://webservices.daehosting.com/" temperature"> ,
<tem:FahrenheitToCelsius xmlns:tem"> ];
```

and, then, can be assigned to a variable by using the associated name, e.g. `reqConv := litConv;`.

Similarly, also partner links are typed in WS-BPEL and un-typed in `Blite`. Therefore, except for the partner links used by the process to interact with its clients, that are automatically generated and typed by `BliteC`, the type of the other partner links must be defined within the PARTNERLINKS block. Each declaration has the following form:

```xml
PARTNERLINK { TYPE => partner_link_type; MY_ROLE partner1; port1_type1; PARTNER_ROLE partner2 => port2_type2; }
```

where the association for `MY_ROLE` can be omitted whenever the process does not play any role. Moreover, to decouple the `Blite` operation names from the WS-BPEL ones, associations of the form (`bliteOperation` => `wsbipelOperation`) may be specified after the definitions of the two roles.

### 4. BLITEC: FROM BLITE TO WS-BPEL

In this section we present the architecture of `BliteC` and explain the correspondence between the `Blite` constructs and the WS-BPEL activities.

#### 4.1 BliteC architecture

`BliteC` is developed in Java\(^2\) to guarantee its portability across different platforms, to exploit the well-established libraries for generating parsers and for manipulating XML documents, and because Java is the reference language for the applications designed around WS-BPEL. Besides the standard Java libraries, we have used JDOM [12] for creating and managing XML documents, JavaCC [11] for generating the parsers that validate the input documents, and JJTree\(^3\) for allowing the parsers to build parse trees (already arranged to support the Visitor design pattern [19]).

The architecture of `BliteC` is graphically depicted in Figure 3. The tool is composed of five main components:

2`BliteC` is a free software; it can be downloaded from http://rap.dsi.unifi.it/blite and redistributed and/or modified under the terms of the GNU General Public License.

3JRE and JDK version 6.

4JJTree is included within JavaCC.
Mapper parses the declarative part of the input Blite program and initializes a map that associates each declared object (e.g. partner link, literal, variable, ...) to its name;

Blite parser analyzes the Blite specification within the input program, completes the map created by Mapper and creates the parse tree of the Blite specification;

WS-BPEL and WSDL generators use the data produced by the above components to generate a WS-BPEL process and the associated WSDL document;

Deployer generates the deployment descriptor and packages all created documents into a deployable file; it is the only 'engine-dependent' component.

Any text editor can be used to write Blite programs. Anyway, to simplify the task, we provide users with a customized version of jEdit\(^5\) [2] equipped with specific features supporting programming in Blite, such as syntax highlighting, auto indentation and direct compiling. The files for the customization can be downloaded with the BliteC distribution archive. The main advantage of jEdit with respect to more professional development environments is that it is multi-platform and lightweight. We are also implementing a development environment with similar features written in Java. Figure 4 shows a screenshot of our environment. In addition to the functionalities of the customized version of jEdit, our dedicated environment also provides text auto-completion, highlight of search results, local deploy and undeploy.

4.2 From Blite to WS-BPEL

We now provide some insights about the transformation of Blite constructs into WS-BPEL activities. Since the same WS-BPEL program might have different behaviours on different engines [25], the translation described here is targeted to a specific engine, i.e. ActiveBPEL. If one want to produce packages intended to be executed by other WS-BPEL engines, the translation has possibly to be properly tailored. Since there is no precise description of the behaviour of the ActiveBPEL engine, it cannot be formally proved that the semantics of the WS-BPEL program resulting from a translation conforms to that of the original Blite program. However, since Blite is a ‘sort of’ lightweight variant of WS-BPEL, the translation we define is quite intuitive and direct, which makes us confident that the original semantics is obeyed. This is of course witnessed by all the experiments we have done.

Communication activities, invokes and receives, are translated in a different way depending on their arguments and their position in the code. Therefore, the translation of Blite programs proceeds in a top-down fashion and, in doing so, the WS-BPEL generator exploits the information previously collected by the Mapper and the Blite parser. For example, as shown in Table 1, if a receive activity is positioned within a pck construct it is translated as an <onMessage> activity; if it is positioned after an invoke (in case of a request-response interaction) it is translated as a synchronous <invoke>; otherwise, it is simply translated as a <receive>. In addition to the excerpts shown in the table, the translation also settles the following activity attributes. If a receive is a start activity, to allow the process to be instantiated, the createInstance attribute must be set to yes. Moreover, if some correlation variables are involved, the corresponding correlation set (whose declaration is generated during the translation of the deployment term) must be specified as a further argument of the <receive> activity. The correlation attributes initiate and pattern are set according to the type of the interaction.

The invoke activity is translated similarly, as shown in Tab-
ble 2; in particular, when it is used in a request-response interaction to send the response, it is translated as a <reply> activity. The translation of the remaining basic activities, as shown in Table 3, is straightforward. In particular, an assign activity involving message variables is translated by possibly using XPATH queries and by exploiting the type of the involved variables (defined in the declarative part) to identify the corresponding parts. Also the translation of the structured activities does not require a significant effort, as shown in Table 4. Finally, as shown in Table 5, a Blite service is rendered as a scope, where the compensation handler is removed and the tag <scope> is replaced by <process>.

5. BLITEC AT WORK

In this section, we present an application of BliteC to some illustrative practical scenarios. The WS-BPEL and WSDL files of the presented services are reported in [16], while all Blite programs and the corresponding WS-BPEL packages can be retrieved along with the BliteC distribution archive.

5.1 A virtual credit card service

A virtual credit card is a prepaid non-physical credit card devised for safe online shopping. A Blite specification for creating and handling a virtual credit card is the following:

```plaintext
s_vcard ::= [ seq
  rcv <p_createcard> o_newcard <x_id,x_amount>;
];
```

A new card is created by invoking the operation o_newcard and specifying a card identifier and the initial amount. The created instance allows the card holder to perform withdrawals by repeatedly invoking the request-response operation o_getcash until the card is empty. For each withdrawal request, the money availability is checked and a message, stored in x_resp, is sent back. The fact that the invoke activity used for the reply is performed along the same operation of the second receive indicates that the two activities form a synchronous request-response interaction, hence the invoke will be translated into a <reply> activity. The card identifier, stored in x_id, is used as a correlation value.
Since the above service does not need to invoke other services, only its address and variables are explicitly declared:

```xml
<Imports>
  <ns0:import location="virtualcard.wsdl" myRole="p_createcard"/>
  <ns0:import location="virtualcard.wsdl" myRole="p_vcard"/>
</Imports>
```

To compile this Bűte program, we have to save the above code into a file (named, e.g., vcard_service.bl) and execute the command java -jar blite.jar vcard_service.bl. This way, the file virtualcardProcess.bpr, which is a WS-BPEL package directly deployable into ActiveBPEL, is generated. Of course, the same actions can be also performed by using the editor or the development environment mentioned in Section 4.1.

The WS-BPEL file included in the generated package, where irrelevant details have been omitted, is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<process name="virtualcardProcess" ... />
<import location="virtualcard.wsdl" ... />
<partnerLinks>
  <partnerLink name="cltPL" partnerLinkType="mwl:cltPLT" myRole="p_vcard"/>
  <partnerLink name="p_createcardPL" partnerLinkType="mwl:p_createcardPLT" myRole="p_createcard"/>
</partnerLinks>
<variables>
  <variable name="var1" messageType="mwl:withdrawalReq"/>
  <variable name="var2" messageType="mwl:withdrawalResp"/>
</variables>
<correlationSets>
  <correlationSet name="x_idCorr" properties="mwl:x_idProp"/>
</correlationSets>
<faultHandlers>
  <sequence> <catchAll /> <empty /> </sequence>
</faultHandlers>
<sequence>
  <receive partnerLink="p_createcardPL" operation="o_newcard" variable="var0" createInstance="yes"/>
  <correlations>
    <correlation set="x_idCorr" initiate="yes"/>
  </correlations>
  <assign>...
  <copy>
    from variable="var0" part="id" 
    to variable="var1" part="id"
  </copy>
  <copy>
    from variable="var0" part="id" 
    to variable="var2" part="id"
  </copy>
  <assign>...
  </sequence>
  <while condition="$var0.amount &gt; 0">
    <sequence>
      <receive partnerLink="cltPL" operation="o_getcash" variable="var1"/>
      <correlations>
        <correlation set="x_idCorr" initiate="no"/>
      </correlations>
      <assign>...
      <if>
        $var0.amount &gt;= $var1.wdrAmount
      </if>
    </sequence>
  </while>
</process>
```

As expected, the resulting WS-BPEL code is more verbose and intricate than the Bűte one. However, the performed translation turns out to be quite ‘clean’, in the sense that each BűteC activity has been translated into the corresponding WS-BPEL one without introducing ‘junk’ code.

To deploy the file virtualcardProcess.bpr, it is sufficient to move it into the engine’s deployment directory bpr. Then, to check that the deploy succeeded, we can use the ActiveBPEL’s administration console that can be accessed by using any browser at the address http://XXX:8080/BpelAdmin (where XXX is the server’s address where the ActiveBPEL engine is running). By selecting Deployed Processes from the menu on the left-hand side, we obtain the list of the deployed processes (Figure 5) among which virtualcardProcess should appear. Now, by selecting Deployed services, we can retrieve the URLs of the two WSDL files corresponding to the partner links for interacting with the service:

http://XXX:8080/active-bpel/services/p_createcardService?wsdl
http://XXX:8080/active-bpel/services/p_vcardService?wsdl

To test the service behaviour, we can use a tool for automatic generation of web service requests, as e.g. soapUI [13], and invoke the service by sending the following SOAP messages:

```xml
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:vir="http://virtualcard/virtualcard.wsdl">
  <soapenv:Header>...
  <soapenv:Body>
    <vir:x_amountE> 100 </vir:x_amountE>
    <vir:x_idEL> 1234 </vir:x_idEL>
  </soapenv:Body>
</soapenv:Envelope>
```

6In fact, when provided with a WSDL file, ActiveBPEL produces as many WSDL files as the different partner links.

Applied Computing Review 34
Processes Euros. In response, the system returns the amount accepted, while the second message is a request for withdrawing 50 Euros accepted and, by selecting Active Processes from the console menu, we can verify that the card instance is still running. If we resend the request we obtain the same response, but the instance status changes to completed.

The first message creates a virtual credit card identified by 1234 with 100 Euros as initial amount, while the second message is a request for withdrawing 50 Euros. In response to the second message we get the string "Response: 50 Euros accepted". The process is initialized by invoking the synchronous request-response operation init. Then, the created instance asynchronously invokes the partner server and, once receives the reply message, appends it at the response and sends the obtained message to the initiator. Notably, in Böte synchronous and asynchronous interactions are rendered in a similar way (i.e. as pairs of activities inv-rcv and rcv-inv); they are distinguished only by the fact that synchronous interactions use the same operation for invoking and receiving.

The declarative part of the asynchronous invoker is:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?blm
    ADDRESSES {
        myns => "http://asyncComm";
        myaddress => "http://XXX:8080/active-bpel/services";
    }
    IMPORTS {
        asS => "http://asyncComm/asyncServer.wsdl";
    }
    VARIABLES {
        <y_id> => asS:req;
        <y_id,y_msg> => asS:resp;
        <y_id,y Resp> => gen:resp,<id,msg>,<xsd:int,xsd:string>;
    }
    PARTNERLINKS {
        PARTNERLINK {
            TYPE => asS:serverPLT;
            MY_ROLE client => asS:clientPT;
            PARTNER_ROLE server => asS:serverPT;
        }
    }
}</?blm
```

The types of the messages corresponding to the asynchronous invocation and the related response are imported by the WSDL document of the asynchronous server (identified by the prefix asS). Moreover, while the partner link to interact with the initiator partner client is automatically generated by BöteC, the partner link to interact with server must be explicitly declared (the types of the partner link and the involved ports are again imported from the server’s WSDL document). Hence, the server must be compiled first.

We report here an excerpt of the WS-BPEL program corresponding to asyncServer

```xml
<payload name="asyncServerProcess" ... >
...
</payloadLinks>
<partnerLink name="serverPL" partnerLinkType="mwl:serverPLT" myRole="server" partnerRole="client" initializePartnerRole="no" />
</partnerLinks>
<variables>
    ... </variables>
<correlationSets>
    <correlationSet name="x_idCorr" properties="mwl:x_idProp" />
</correlationSets>
```

Since this service does not need to invoke other services, the declarative part is just as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<process name="asyncServerProcess" ... >
...
</payloadLinks>
<partnerLink name="serverPL" partnerLinkType="mwl:serverPLT" myRole="server" partnerRole="client" initializePartnerRole="no" />
</partnerLinks>
<variables>
    ... </variables>
<correlationSets>
    <correlationSet name="x_idCorr" properties="mwl:x_idProp" />
</correlationSets>
```

Although WS-BPEL provides the means for implementing asynchronous communication, it requires programmers to directly deal with endpoint references, as shown by the example in Section 2. Instead, in Böte this relevant communication pattern can be easily and transparently implemented.

A Böte specification for receiving a request and asynchronously replying with the string Hello is the following:

```xml
asyncServer := { s_asyncServer }{ x_id };;
rcv <server,client> request <x_id>;
    inv <server,client> request <x_id>;
    inv <client> response <x_id,x>
    x := "Hello";
    inv <client> response <x_id,x>
    qes ];;
asyncServer := { s_asyncServer }{ x_id };;
```

The corresponding invoker service is rendered in Böte as

```xml
s_asyncClient := { s_asyncClient }{ y_id };;
inv <y_clt> init <y_id,y_resp>
    y_resp := "Response: ".y_msg;
    rcv <client> response <y_id,y_msg>;
    y_resp := "Response: ".y_msg;
    inv <y_clt> init <y_id,y_resp>
    qes ];;
asyncClient := { s_asyncClient }{ y_id };;
```

5.2 On asynchronous communication

Frequently, it is assumed that a service request can be processed in a reasonable amount of time, justifying the requirement that the invoker waits for a response related to a synchronous request-response operation. In a business process setting, where communication costs are high or network latency is unpredictable, such assumption usually does not hold and the interactions are better modeled by asynchronous message exchanges. Therefore, an asynchronous messaging approach is considered good practice for web services and service orchestrations in particular.

A Böte specification for receiving a request and asynchronously replying with the string Hello is the following:

```xml
asyncServer := { s_asyncServer }{ x_id };;
rcv <server,client> request <x_id>;
    inv <server,client> request <x_id>;
    inv <client> response <x_id,x>
    x := "Hello";
    inv <client> response <x_id,x>
    qes ];;
asyncServer := { s_asyncServer }{ x_id };;
```

The process is initialized by invoking the synchronous request-response operation init. Then, the created instance asynchronously invokes the partner server and, once receives the reply message, appends it at the response and sends the obtained message to the initiator. Notably, in Böte synchronous and asynchronous interactions are rendered in a similar way (i.e. as pairs of activities inv-rcv and rcv-inv); they are distinguished only by the fact that synchronous interactions use the same operation for invoking and receiving.

The declarative part of the asynchronous invoker is:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<process name="asyncServerProcess" ... >
...
</payloadLinks>
<partnerLink name="serverPL" partnerLinkType="mwl:serverPLT" myRole="server" partnerRole="client" initializePartnerRole="no" />
</partnerLinks>
<variables>
    ... </variables>
<correlationSets>
    <correlationSet name="x_idCorr" properties="mwl:x_idProp" />
</correlationSets>
```
We show here an application of BìteC to a scenario built upon the auction service drawn from the WS-BPEL specification document and already introduced in Section 2.

The auction service is defined in Bìte as:

```blm
s_auction ::= {s_auction}{x_auctionId};;

<sequence>
<receive partnerLink="serverPL" operation="AktionsID" variable="var0">;
<assign> var0 := "someId";
<invoke partnerLink="serverPL" operation="response" var0="someId">;
</sequence>

<auction ::= {s_auction}{x_auctionId};;

<sequence>
<assign> x_auctionId := "someId";
<invoke partnerLink="serverPL" operation="response" x_auctionId="someId">;
</sequence>

To enable asynchronous communication, the translation automatically puts an additional receive activity (highlighted by a gray background) in the generated WS-BPEL code. This activity will be used by clients to communicate their addresses, which are then assigned to the partner link used for the callback operation. Similarly, in a transparent way, BìteC equips the clients invoking this partner link with the symmetric invoke activity.

5.3 An auction service scenario

We show here an application of BìteC to a scenario built upon the auction service drawn from the WS-BPEL specification document and already introduced in Section 2.

The registration service is rendered in Bìte as follows:

```blm
register ::= {register}{x_registrationId};;

<sequence>
<assign> x_registrationId := "someId";
</sequence>
```

Here, differently from the program in Section 2, the endpoint references of the registration, buyer and seller services are not explicitly handled by the programmer. Moreover, the only partner link declared is that used for interacting with the registration service.

The registration service is rendered in BìteC as follows:

```blm
register ::= {register}{x_registrationId};;

{ seq
  rcc <register,auction> process <x_id,x_amount>;
  inv <auction> regAnswer <x_id,x_registrationId>
  myaddress => "http://localhost:8080/active-bpel/services";
}<sequence>
```

Its behaviour is very simple: the process gets instantiated by the auction service by invoking the operation process; then, the created instance replies with a registration identifier. For the sake of simplicity, we do not model here the generation of a unique identifier (which most likely could be provided by another service).

Finally, we report below the seller service (the buyer service is defined similarly):

```blm
v_seller ::= { v_seller }{ x_id };;

{ seq
  rcc <initSeller,initiator> init <x_id,x_cc,x_shipCost>
  rcc <seller> answer <x_id,x_resp>
  myaddress => "http://localhost:8080/active-bpel/services";
}<sequence>
```

The above specification is very similar to that of the asynchronous client described in Section 5.2.

Once the above programs have been compiled and deployed, regardless of the activation order of the buyer and seller, both services will receive the message 'Thank you!' indicating the successful termination of the orchestration.
5.4 An auction service scenario with fault and compensation handling

We now extend the scenario introduced in Section 5.3 with fault and compensation handling. Basically, we allow the buyer and the seller services to cancel the auction, which causes its unregistration. Thus, the auction service becomes:

```plaintext
invReg :=
  seq
  inv <register.auction> process <x_auctionId,x_amount>;
  rcv <auction> regAnswer <x_auctionId,x_registrationId>;
  flw
  inv <seller> answer <x_auctionId,x_resp> |
  inv <buyer> answer <x_auctionId,x Resp> wlf qes;;
compReg :=
  seq
  x_motivation := "Auction cancelled";
  inv <registerComp> unregister <x_auctionId,x_motivation> qes;;
fh :=
  seq
  x_resp := "The registration has been cancelled";
  flw
  inv <seller> answer <x_auctionId,x_resp> |
  inv <buyer> answer <x_auctionId,x Resp> wlf qes;;
s_auction :=
  [ seq
    flw
    rcv <auctionS,seller> submit
    <x_auctionId,x_creditCardNumberS,x_shippingCost> |
    rcv <auctionB,buyer> submit
    <x_auctionId,x_creditCardNumberB,x_phoneNumber> wlf
    x_amount := 1;
    x Resp := "Thank you!";
    [invReg @ empty * compReg] rcv <auctionCancel> cancel <x_auctionId> throw
  qes $ fh]];
```

The previous examples show how BliteC can be used to generate complete orchestration scenarios where each service is obtained by a Blite specification. However, our tool can be also used to orchestrate Blite services together with non-Blite ones. To this aim, our specification language provides two simple constructs, get and set, that permit manipulating XML messages exchanged with (possibly) non-Blite services.

Consider a service that receives a US zip code and a preference for the temperature scale, i.e. either the character c for Celsius or f for Fahrenheit, contacts a first service for getting the current weather conditions and the temperature in degrees Fahrenheit of the city corresponding to the zip code, possibly contacts a second service for converting degrees Fahrenheit into degrees Celsius, and finally sends the obtained weather and temperature information to the invoker. To implement this service we have orchestrated two web services freedly provided by CDYNE [1]. The corresponding Blite specification is as follows:

```plaintext
zipWeatherClient :=
  {{ seq
    rcv <initWeatherClient,initiator> init <num,x,scale,reqWeather>;
    inv <weather, ch_weather> reqWeather <reqWeather>;
    rcv <cb_weather> weather <respWeather>;
    x := get(respWeather, /wth:GetCityWeatherByZIPSoapOut);
    x_city := get(respWeather, /wth:GetCityWeatherByZIPSoapOut /wth:Description);
    x_temF := get(respWeather, /wth:GetCityWeatherByZIPSoapOut /wth:Temperature);
  if (scale == "c")
    [seq
      reqConv := newConv;
      set(reqConv, /temp:FahrenheitToCelsius /temp:FahrenheitToCelsiusResult x_temC := convert_cb(reqConv, x_temF);]
    set(x_temC, /temp:CelsiusToFahrenheit /temp:CelsiusToFahrenheitResult x := x_temC * 9/5 + 32)
  else (x := x_temF * 9/5 + 32)
  inv <initiator> init <num,x> qes $ num]};
```

5.5 Orchestrating non-Blite services

Once the auction service receives the two requests, it executes the scope activity [invReg @ empty * compReg]: the primary activity invReg interacts with the registration service and then replies to buyer and seller, while the activity compReg is the corresponding compensation handler that simply invokes the registration service to cancel the registration. After the scope completes, the auction service waits for a cancellation message by either the buyer or the seller. The reception of such a message generates a fault (through the activity throw). The fault is handled by the fault handler fh that automatically activates the compensation handler compReg (we refer to [25] for a complete account of the default compensation mechanism) and sends two messages to buyer and seller to notify that the registration has been cancelled.
The \texttt{get} construct is used here to extract information from the XML messages received from the two external web services, while the \texttt{set} construct is used to insert the received temperature expressed in degrees Fahrenheit into the XML request message for the converter service, whose structure has been previously initialized by means of the literal \texttt{litConv}. Once the program has been compiled and deployed, if we invoke the operation \texttt{init} by specifying the zip code 90210 and the scale Celsius, we will get back a string of the form:

\begin{verbatim}
The current weather at Beverly Hills is: Clear, the temperature is: 14.444444 °C
\end{verbatim}

6. CONCLUDING REMARKS

We have presented \texttt{BliteC}, a software tool for supporting a rapid and easy development of WS-BPEL applications. The tool aims at solving some well-known programming problems of WS-BPEL caused by its XML syntax, lack of a formal semantics, and non-standardization of the deployment procedure. Basically, \texttt{BliteC} takes as inputs programs written in \texttt{Blite}, a prototypical orchestration language inspired to WS-BPEL but with a simpler syntax and a well-defined operational semantics, and provides as output the corresponding deployable WS-BPEL programs.

The aim of facilitating the development of WS-BPEL applications is shared also by the several graphical editors that permit designing WS-BPEL processes, among which we mention the designers embedded in Oracle BPEL Process Manager [5], Intalio|Designer [10], ActiveVOS Designer [7], and Eclipse BPEL designer [9]. Although their use is quite intuitive, developing large applications by using them can be awkward and annoying compared to the more classic textual approach. Indeed, graphical notations turn out to be suitable for beginner WS-BPEL programmers to represent simple business process workflows, but do not allow more expert programmers to exploit commonly used functionalities, such as e.g. copy/cut/paste, and are inappropriate for expressing some (textual) information, such as e.g. correlation sets. Moreover, graphical designers have a significant negative impact on performance during the programming phase (that is, indeed, the phase of the software development process on which we focus on), since they usually are plugins of heavy software development environments such as JDeveloper [3] and Eclipse [4]. Some other works with a similar aim are [28, 32, 14]. The first two present some tools that produce WS-BPEL processes starting, respectively, from UML and Petri Nets-based representations of SOC applications. Due to the use of graphical representations, also these tools suffer from the problems previously mentioned. Instead, the third one proposes a mapping from a pi-calculus based formalism into WS-BPEL. In all three approaches, only non-executable WS-BPEL processes are generated, i.e. the generated code should be thought of as a template code where, besides binding and deployment details, programmers have also to define things such as partner links, variables, port types, correlations sets, etc. by editing the generated files. Another related work is [29], which proposes a different approach to develop SOC applications that still relies on a formal language. However, input programs are directly executed in a purposely developed engine, rather than being translated into and deployed as WS-BPEL processes.

Currently, the WS-BPEL packages generated by \texttt{BliteC} are intended to be deployed on ActiveBPEL. This is just to demonstrate feasibility of our approach. In fact, \texttt{BliteC} has been designed so that the generation of deployment descriptors for different engines can be easily integrated, and we plan to enable it to produce packages also for other freely available engines, such as Oracle BPEL Process Manager, Apache ODE [8] and Beepell [23]. Of course, to preserve the semantics of the original \texttt{Blite} programs, one has to study the inner implementation of every supported engine and to define a customized translation. Since no engine has a formal description of its behaviour, this study has to be carried out by means of experimental tests and, most of all, no formal proof of semantics preservation can be done. It is also worth noticing that the semantics of \texttt{Blite}, which is in fact quite close to that of ActiveBPEL, could be rather tough to render by some WS-BPEL engines, whose semantics may significantly differ on low-level implementation details (e.g. message queue handling) or may more strictly (or inappropriately) enforce some WS-BPEL constraints. For instance, it may happen that a process instance should receive a message from a partner according to the \texttt{Blite} semantics, while instead the message cannot be effectively accepted according to the semantics of the considered engine, due to e.g. some peculiar correlation constraint. In such a case, \texttt{BliteC} should identify the potential conflicting receives in the generated WS-BPEL program and, e.g., replace them by a single receive enabling some proper coordination activities.

We also plan to enrich the \texttt{BliteC} development environment presented in Section 4.1 with further functionalities, such as deployment/undeployment facilities over remote servers, and debugging tools, such as automatic generation of web interfaces for invoking the created services and log recording based on an embedded dedicated web service. This latter tool could require to extend the syntax of \texttt{Blite} accepted by \texttt{BliteC} with a construct for printing strings into the log that would be translated into a WS-BPEL one-way interaction.

For what concern the language \texttt{Blite}, we also intend to investigate its extension to cover some WS-BPEL constructs...
that at the time being have been left out, such as timed activities, event and termination handlers, and more sophisticated forms of fault and compensation handling involving named faults and compensation of specified scopes. We do not envisage any major issue in translating such constructs in WS-BPEL code, while their addition to B\text{\textae} would require to significantly revise the formal definition of the operational semantics of the language.

Finally, we intend to develop formal analysis techniques, e.g. based on model checking (as in [14, 15]), for B\text{\textae} specifications. This way, we would be able to specify in B\text{\textae} an orchestration scenario, validate its behaviour by using formal tools, and deploy it as a set of WS-BPEL programs.

7. ACKNOWLEDGMENTS

We thank Alessandro Lapadula for his fundamental contribution to the definition of the language B\text{\textae}. We also thank the anonymous reviewers of SAC’10 and Applied Computing Review for their helpful comments, and people attending the SOAP track at SAC’10 for the fruitful discussions following the presentation of our work. Finally, we thank the chairs of the SOAP track for their invitation to submit to the Applied Computing Review and their support.

This work has been sponsored by the EU project SENSORIA, IST-2005-016004.

8. REFERENCES

About the authors:

Luca Cesari is a student in Scienza e Tecnologia dell'Informazione at the University of Florence. He received a Master degree in Computer Science from the University of Florence (April 2009) with the thesis "Design and implementation of a tool for supporting the development of WS-BPEL applications" supervised by Prof. Rosario Pugliese and Francesco Tiezzi. His current interests focus on type systems, model checking and service-oriented systems.

Rosario Pugliese received a Laurea degree cum laude in Scienze dell'Informazione from the University of Pisa (April 1991) and a Ph.D. degree in Computer Science from the University of Rome 'La Sapienza' (November 1996). Pugliese is with the Dipartimento di Sistemi e Informatica. He is associate professor of Computer Science at the Facoltà di Scienze Matematiche Fisiche e Naturali (School of Science) of University of Florence since 2002. He has been assistant professor in the same School since 1999. Previously, he has benefited from research fellowships and contracts funded by the National Research Council (CNR) and by the University of Florence. Pugliese research interests are centered around formal methods and tools (process calculi, behavioural equivalences, type systems, modal logics) for specification and verification of concurrent and distributed systems. His current research focusses on Global Computing and Service Oriented Computing.

Francesco Tiezzi received a Laurea degree cum laude in Informatica from the University of Florence (January 2005) and a Ph.D. degree in Computer Science from the same university (April 2009), under the supervision of Prof. Rosario Pugliese. Previously, during 2005, he had a Post-degree collaboration with the Dipartimento di Sistemi e Informatica of the University of Florence. Since 2009, Tiezzi is a Post-doc at the same department. Tiezzi's research interests include semantics of languages for Service-Oriented Computing, process calculi, type systems, models and tools supporting specification and verification of service-oriented systems, and web-based programming languages.
Register-Relocation: a Thermal-aware Renaming Method for Reducing Temperature of a Register File

Jungwook Kim  
Department of Computer Science and Engineering  
Seoul National University  
Seoul 157-44, Korea  
kelvin@snu.ac.kr

Seong Tae Jhang*  
Department of Computer Science  
The University of Suwon  
Gyeonggi-do 445-743, Korea  
stjhang@suwon.ac.kr

Seung-Jin Moon  
Department of Computer Science and Engineering  
The University of Suwon  
Gyeonggi-do 445-743, Korea  
sjmoon@suwon.ac.kr

Chu Shik Jhon  
Department of Computer Science and Engineering  
Seoul National University  
Seoul 157-44, Korea  
csjhon@cse.snu.ac.kr

ABSTRACT
The manufacturing process of microprocessors becomes increasingly fine and the clock frequency is rapidly growing. Since the corresponding power consumption, however, is not reduced, power density is increased dramatically. The generated heat by the power density induces high temperature. The high temperature causes many problems: calculation errors, aging, leakage power, and cooling costs. Register file produces the highest temperature in a microprocessor because of extremely high access frequency and its small area. We demonstrated that the existing renaming unit causes high temperature since it allocates registers imbalanced. Our idea is to redistribute evenly register allocations and accesses across the whole range of the register file; consequently, the overall power density is reduced and then the temperature is lowered. The proposed method can be implemented by adding a small logic to the traditional renaming unit with around 1.5% overheads. As a result, temperature drop was up to 10.4°C (11%) on average 4.6°C (6%). We also achieved leakage power savings and performance improvements by the temperature drop.

Categories and Subject Descriptors
C.1.0 [Processor Architectures]: General

General Terms
Design, Experimentation

Keywords
Register File, Temperature Reduction, Register Relocation, Embedded Processor

1. INTRODUCTION
In recent years, it has been able to integrate a great number of transistors to limited space on the chip due to development of semiconductor manufacturing technology. Accordingly, current microprocessors show more features and faster performance in spite of the smaller area compared to the older microprocessors. While the physical size of the microprocessors becomes progressively smaller, the power consumption requirements are rarely reduced because the software increasingly demands high performance. Thus, the power density on the microprocessor is rapidly increased, and the high temperature caused by this increase brings about many problems: malfunction, poor durability, increased leakage power, and cooling costs. Low-power processors used in embedded devices are not free from the thermal problems as well as high-performance microprocessors for servers; moreover, this issue may be intensified in the future. For this reason, many studies have been introduced to lower the temperature in microprocessors.

Among the studies, the most common technique is the Dynamic Thermal Management (DTM). The technique adjusts the operating frequency or the voltage of a microprocessor, or it regulates the instruction-fetch to control the temperature. The most common mechanism is as follows. The processor temperature is monitored in real-time. If the temperature rises above a predetermined threshold temperature, the operating frequency or the voltage of the processor is scaled, or the instruction-fetch is suspended until the temperature goes down to a safe point below the emergency temperature. As the instruction pipeline is slowly performed or suspended by this control, the power consumption of each unit in the microprocessor is dramatically reduced and then the temperature is decreased. When the temperature goes down to a safe area below the critical point, the clock frequency or the voltage is increased again or the instruction-fetch is restarted to operate in normal speed. Since the DTM is such an effective dynamic temperature control, a number of different methods have been introduced and studied. For example, Skadron et al [1] proposed various DTM techniques: “temperature-tracking” frequency scaling, localized toggling, and migrating.

While the DTM is widely applied in many places and many studies are in progress, the performance loss is inevitable. The more frequent the temperature-controlled intervention becomes, the slower the microprocessor is. As DTM is a piece of software at the expense of the performance, the need for research about that have to adopt a temperature resistant structure to suppress high temperature from the design stage of a microprocessor has been emerged. In terms of the microprocessor architecture, the register file shows the highest power density as it has a severe access frequency and occupies a relatively small area. As a result, the register file is known to the hottest unit in the microprocessor. For this reason, several studies have been introduced to reduce the power density of the register file [2][3][4]. Among the studies, especially, we paid attention to Zhou et al [3]. They showed that

*corresponding author
what makes the register file the hottest unit is mostly due to the highly clustered register file accesses where a set of few registers physically placed close to each other are accessed with very high frequency, and they proposed a compiler-based register re-assignment technique which purpose is to break groups of registers and to uniformly distribute the accesses to the register file. We were interested in such information and then found that accesses and allocations of the register file actually were not conducted uniformly through a preliminary investigation. Furthermore, we demonstrated that the method of the traditional renaming unit caused the non-uniformity. Figure 1a shows this uneven distribution of register accesses and allocations. The x-axis corresponds to the entry number of the register file.

Typically, the operation of a renaming unit is as follows. If the output dependency is not resolved when the register-writing is executed, the renaming unit allocates a new free entry of physical registers to the register scheduled to be written. However, as can be seen from Figure 1a, the highly clustered accesses were found on the only one side of the register file by our preliminary experiments. This can be seen that the existing renaming unit was not designed to consider the power density and temperature. Figure 1b presents an example of the emergency temperature due to this highly clustered pattern on the right side.

In this paper, we propose an idea that evenly redistributes accesses to the full range of the register file through the improvement of the traditional renaming unit. The idea can be implemented by attaching only a small logic to the traditional renaming unit. The detailed method and operation of the attached logic will be described in Section 2.2. If the redistribution lowers the power density of the register file, the temperature will be reduced. The temperature drop of the hottest unit may bring a positive effect to lower the entire temperature of the microprocessor. In addition, many benefits are expected due to the decreased temperature. First, leakage power savings are expected. Then, the performance improvement is expected since the performance loss by DTM-intervention will be disappeared because many hotspots can be removed if the temperature decreases to a safe area below the emergency point. Last of all, the chip durability may be less affected and the cooling costs savings are expected.

The proposed method by us has some distinctive advantages in comparison with conventional techniques. As our technique is not a static method dependent on the compiler but a dynamic scheme using hardware logic, the development potential remains. By adding some functions to our logic in the future, it can respond to thermal behavior in real-time according to the operating characteristics of the program. For example, by such a fixed way of a static method based on the compiler or lowering the power density by changing the placement of wires connected to the register file, it will be impossible to cope with the malicious attack such as the thermal virus. The malicious code can cause even greater problems if it attacks by taking advantage of the fixed re-assignment structure. In contrast, our method could be the favorable way to the malicious code as the functionality of our logic can be improved to adequately respond while the program behavior is monitored in real-time.

2. REGISTER RENAMING

2.1 Conventional Renaming Method

Generally, modern superscalar processors have 32 architectural registers and more than 32 physical registers. For example, in Alpha 21264 processor, which has been representative as a high-performance microprocessor and was selected for our experimental target, the physical integer register file has 80 entries. Its higher part (i.e. 0–39 entry number) consists of 32 architectural registers and 8 shadow registers, and the other part (i.e. 40–80 entry number) is used as physical registers for the allocation of writing.

The register renaming is a technique which preserves “program order” against data hazards; the hazards occur among two instructions using the same register or memory location. For example, consider two instructions $I_1$ and $I_2$; $I_1$ and $I_2$ are “load r1, r2, r3” and “add r2, r4, r5”, respectively; $I_1$ occurring before $I_2$ in program order. Commonly, a less time-consuming instruction is executed before a long-time instruction in an Out-of-Order microprocessor. Since the “load” instruction of $I_1$ takes more time than the “add” instruction of $I_2$, $I_2$ will be executed before $I_1$. Thus, $I_2$ may try to write r2 register before it is read by $I_1$, so $I_1$ incorrectly gets a new value (i.e. WAR hazard is occurred). In such a case, the renaming logic assigns the second r2 register to a free physical register within the range of 40–80 entry numbers;
hence, $I_2$ writes the value to a new destination and $I_1$ gets the correct value. Consequently, the program order is preserved by the renaming technique.

However, from our simulation, we found that the most assignments by the conventional renaming scheme clustered on one side of the register file, and it caused high temperature. Since the assignments for register writing are fixed to the only half range (i.e. 40–80 entry number), and accesses for writing hold a considerable portion in the total accesses, the power density in the right side of the register file is rapidly increased. As a result, the maximum temperature appears in that region; moreover, the hotspots (i.e. emergency temperature points) are also created. Figure 1a shows an example of the concentrated accesses on the half of the physical registers in gzip program; the x-axis represents an entry number of the physical registers. Figure 1b shows an example of the thermal map (e.g. gzip); the steady-state temperature rises to the hotspot point (e.g. 90°C) in the right part of the registers.

On the other hand, if the renaming unit assigned a register with a consideration of power density, the temperature of the register file would be decreased, and the number of DTM interventions also would be reduced. In other words, if the allocations for register writing were distributed uniformly to the full range of the register file, then the writing accesses also were distributed uniformly; consequently, the power density and temperature would be decreased and the performance loss also would be reduced.

2.2 Proposed Method: Register Relocation

The goal of our idea is the reduction of the temperature in a register file. The goal may be achieved by uniformly distributing accesses throughout the full-entries of the registers. Our idea is a re-mapping technique revealing that architectural registers (i.e. entry number 0–40) are relocated to the full range of entry numbers (i.e. 0–79) with only the even number allocation, and also that the assignments to physical registers (i.e. 40–80) are also repositioned throughout whole register file area (i.e. 1–80) with the odd number. Our strategy is as follows:

First, the traditional renaming unit allocates an index number of a physical register entry to an architectural register. Next, a new index number is generated by our simple algorithm: if the index number is less than 40, then a new index number will be obtained from multiplying the first index number by 2; otherwise (i.e. 40–80), we subtract 40 from the first index and multiply the subtracted value by 2, and ‘1’ is added. This simple algorithm can be implemented by a small logic, and the logic can be attached to the traditional renaming unit; the attached logic consists of six small components: an eight bit adder, an eight bit shift register, a comparator, an OR gate, and two 2:1 muxes. Our algorithm can be expressed as follows.

\[
C = \text{Full entry} / 2 \\
if \ i \geq C \ \\
i = 2 \cdot (i - C) + 1 \\
else if \ i < C \\
i = 2 \cdot i
\]

Their detailed operations are described as follows. At first, the comparator checks if the first index number is less than 40 and passes the result to two 2:1 muxes. The adder subtracts 40 from its input data and sends it to the first 2:1 mux. The first 2:1 mux selects an input with the result signal of the comparator and forwards it to the shift register. The shift register does a multiply by shifting only one bit (e.g. 2*x). The OR gate does the ‘+1’ operation within ‘2^i+1’. The OR gate receives two inputs: a multiply result from the shift register and the constant number ‘1’, and then it does the OR operation with the two inputs; the OR gate sends its result to the second 2:1 mux. The second 2:1 mux...
receives two inputs from the shift register directly and the OR gate, and it selects one input with the result signal from the comparator; finally, it forwards the selected input to the output port connected to the next pipeline stage. Figure 2a shows this new logic structure, and Figure 2b describes our mapping scenario; the higher part and the lower part of the original entries are relocated to the even entries and the odd entries, respectively.

3. EXPERIMENTAL RESULTS

3.1 Simulation Parameters

Our simulation environment targets the Alpha 21264 microprocessor. The parameters of the simulation were brought from the Alpha 21264 processor core [5]. Table 1 reports the configuration that was assumed in our simulations.

3.2 Methodology

For more precise simulations, we used Sim-alpha [6] as a main simulator. Since the floorplan file [1] of the Alpha21264 was used in our thermal experiments, we needed more accurate simulations on the Alpha Processor.

The Sim-alpha simulator is based on the SimpleScalar [7], but it simulates the Alpha core more accurately than the SimpleScalar. While the SimpleScalar provides only a Register Update Unit (RUU) integrating many essential units (e.g. renaming logic, issue queue, reorder buffer), the Sim-alpha provides many essential logics individually and then implements them with more detailed behaviors. Especially, it is quite a help to our simulation that the relation between the architectural registers and the physical registers is obviously separated and defined, and the realization of the renaming map table and the reorder buffer is relatively exact, compared with the SimpleScalar.

Our power consumption models referred to the models of Watch [8]. The parameters of our power model followed the 65 nm technology guided by International Technology Roadmap for Semiconductors (ITRS). Our floorplan file for the thermal experiments also was fitted to 65 nm scales; the die size of the microprocessor core is 36 mm².

Temperature simulations were conducted by HotSpot [9] with power trace files which were generated from Sim-alpha and Wattch. For more exact simulations, every power trace file was simulated as twice; at first, the steady-state temperature was obtained by the first thermal simulation, and it was used as the initial temperature for the next phase of the thermal simulation. This two-phase process is a typical method in HotSpot simulator.

We used benchmark programs from SPEC CPU2000 [10]. For the experiments efficiency and reducing the simulation time, we used fast-forwarding data from SimPoint [11]. SimPoint provides fast-forwarding data by analyzing each characteristic of SPEC2000 benchmark programs. In other words the study is that the results of unique segments are little difference with the results of whole sectors. Accordingly, the time of unnecessary experiments can be considerably saved.

<table>
<thead>
<tr>
<th>Table 1. Simulation Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Fetch Queue Size</td>
</tr>
<tr>
<td>Instruction Fetch Queue Width</td>
</tr>
<tr>
<td>Instruction Fetch Queue Speed</td>
</tr>
<tr>
<td>Map(Rename) Width</td>
</tr>
<tr>
<td>Issue Width</td>
</tr>
<tr>
<td>Commit Width</td>
</tr>
<tr>
<td>Branch Predictor</td>
</tr>
<tr>
<td>Reorder Buffer Size</td>
</tr>
<tr>
<td>Load Queue Size</td>
</tr>
<tr>
<td>Store Queue Size</td>
</tr>
<tr>
<td>Issue Queue Size : INT / FP</td>
</tr>
<tr>
<td>L1 I/D Cache</td>
</tr>
<tr>
<td>Fast forwarding</td>
</tr>
<tr>
<td>Clock speed</td>
</tr>
<tr>
<td>Technology</td>
</tr>
</tbody>
</table>
3.3 Results and Discussion

We conducted the experiments by dividing into the integer programs and the floating-point programs, and we summarized the results by measuring the maximum temperatures of the steady-state and the transient state in each result of the program.

First, we get the power consumption file of each unit over time through the main simulator. Next, the results of the temperature-simulation can be achieved by putting the power consumption file and the floorplan file of the microprocessor as the input of the HotSpot simulator. The achieved results consist of the thermal map of the steady-state temperature and the recorded trace-file of the temperature changes over time.

First, we will analyze the results of the integer programs. Then the results of the floating-point programs will be discussed in Section 3.4.

Figure 3 shows temperature changes of each program in the integer register-file. The left two bars present the comparison of the existing method and our method in steady-state temperature, and the right two bars show the comparison in peak temperature. As can be seen in Figure 3, our method shows the same effect on the temperature reduction of the steady-state as well as the peak temperature. Especially, in case of gcc program, the best results are shown as the reduction above 10°C both the steady-state and peak temperature. In whole programs the average reduction of steady-state temperature was 4.6°C, and that of peak temperature was 4.7°C. If the emergency temperature is defined as 80°C, in the steady-state temperatures, four programs are exceeding the emergency point: bzip2, crafty, gcc, and gzip. In the peak temperatures, three programs have been added to the early four programs: gap, parser, and twolf; therefore, total seven programs are exceeding the emergency temperature. Finally it can be seen that the average of the peak temperatures is already exceeding the emergency temperature. In spite of such high peak-temperatures, because our method has the effect of reducing the peak temperature as well as the steady-state temperature, it can be found in the right bars of Figure 3 that the average of the peak temperature has been decreased below the critical temperature.

Figure 4 shows an example that discusses a cause of the temperature reductions in terms of thermal map, and it reveals that the heating pattern of gcc program has been changed. Figure 4a presents the heating pattern which was created by the imbalanced register-allocation of the existing method. Like the example of Figure 1a, it depicts severely increased temperature by high power density of the right part, and a big thermal gradient also may be created by steep temperature differences between the left and the right; it is known that the thermal gradient is harmful to the durability of a circuit. In contrast, as shown in Figure 4b, the maximum temperature of steady-state also has been considerably reduced, and the overall heating pattern is uniform. These results also may be due to our register-relocation method. In addition, it can be expected that such a uniform pattern lowers the likelihood of a steep thermal gradient.

While Figure 4 shows the thermal map of steady-state temperature, Figure 5 shows the temperature changes over time in gcc program. In three-dimensional coordinate of Figure 5, x-axis displays entry numbers of the register file; y-axis shows the execution cycles over time; and z-axis corresponds to the temperatures of each entry. Figure 5a shows the temperature changes of the register file by the existing register assignment method. In this case, it cannot be found that the peak temperature goes down below the emergency point (e.g. 80°C) during whole execution cycles. In previous section of near 300 (x 100k) cycles point, it can be seen that in the right side of the register file, the temperature has been severely increased enough to exceed 100°C; in contrast, temperature of the left side is about 70°C; thus, the temperature difference is more than 30°C between the left side

![Figure 4 Thermal map of steady-state temperature in gcc](image)

![Figure 5 Temperature distribution and change over time in the register file](image)
and the right side. Accordingly, it can be easily expected that the thermal gradient by the sudden difference will be a significant damage to the lifetime of the register file. The maximum of the peak temperatures is kept above the emergency point even in the next section after the 300 (x 100k) cycles point. In that section, also it can be found that the difference is about 30°C compared to the lowest temperature.

Consequently, the existing renaming method causes the continuous high temperature and the extreme difference of temperature. Generally, the thermal gradient by the sudden difference is a cause of the thermal-cycling creation. Rapid thermal-cycling between hot point and cold can damage the chip by giving the thermo-mechanical stress to a circuit [12] [13]. Therefore if programs such as gcc are continuously performed, the microprocessor cannot avoid a severe damage of the register file circuit as well as the performance loss.

On the other hand, in case of our method, Figure 5b shows some stable heating pattern. In section before the 300 (x 100k) cycles, though the maximum temperature is seen above 80°C, the heating-pattern is shown as the uniform distribution compared to Figure 5a. Of course, in such a uniform heating pattern, the created thermal-gradient will be very small enough to ignore the impact to the circuit. Also the maximum temperature of this section has been considerably reduced after the 300 (x 100k) cycles point, and the uniform heating-pattern is being kept. Since the maximum temperature is below the emergency point, the performance loss by DTM-intervention will not occur.

Accordingly, it can be seen that our method has many advantages in the overall comparison. First, the decrease of steady-state and peak temperature leads to relative performance improvements by minimizing DTM; in addition, it is expected to save the leakage power by the reduced temperature. Second, it is expected that the uniform heating pattern contributes to extend the lifetime of the register file circuit by minimizing the thermal gradient creation.

However, the result of “vortex” program is quite different with other programs; the temperature rose instead of falling. Our analysis for this fail is as follows. Two important reasons can be considered. First, in most applications, the access count of the R31 register is a considerable portion in the total accesses; the R31 register is the 32th entry of the register file and used as “zero register”. Second, in vortex, the access count of R31 register is relatively greater than others (e.g. the percentages are 8% and 3% in vortex and gcc, respectively). In other words, the number of accesses of the 32th entry (i.e. R31 register) gives vortex more impact. In such a case, since some frequently accessed entries are relocated near R31 register entry by our proposed method, the power density near R31 register may be inevitably higher than the existing register file compared with other applications. Figure 6a and 6b shows that the position of the maximum temperature region was changed by our method. Also, from Figure 6c and Figure 6d, the change of access-density between the existing method and our method can be observed near R31 register. In such a case, a more dynamic control may be required and for future work, we will study a smarter scheme.

### 3.4 Results of Floating-Point Programs

Figure 7 shows the results of floating-point programs in the integer register file. The left two bars indicate the maximum temperatures of the steady state, and the right two bars show the maximum temperatures of the transient state. As can be seen, the overall resulting temperatures are lower than the integer programs. Since the floating-point programs have the high utilization of the floating-point register file as well as the integer register file, the access to the register file may be distributed on both sides. Also, from the bar 2 and 4, it can be found that the effect is not high though our method has been applied. This may be due to that the overall temperature is low and the emergency situations are hardly ever occurred. An example of this can be seen from Figure 8.

Figure 8 shows the temperature-changes according to the execution cycles of equake which is one of the floating-point programs. As shown in Figure 8, the characteristic of the equake
program is depicted as a unique pattern that the access count to the register file is suddenly increased for short time at regular intervals. In this case, from Figure 8a, also it can be seen that the access are clustered to the right in the existing register file; however, the temperature difference between the clustered right-part and the non-clustered left-part is not greater than the integer programs. Thus, though the imbalanced access have been uniformly distributed by our method, the temperature drop is actually not large; however, though the gap of temperature drop is small, it can be found that the steepness of the thermal gradient is considerably mitigated in Figure 8b compared to Figure 8a. Also such an alleviated gradient may be helpful to the durability of the register file.

In addition, as can be seen in Figure 7, because there is no case of exceeding the emergency point in the peak temperatures, according to this, the performance loss by DTM will not occur. Since this is the results from the integer register-file used by floating-point programs, we will monitor the temperature changes again through the experiments of the floating-point register file for the future work. Also, as shown in the equake result of Figure 7, though the overall steady-state temperature belongs to the very low side, it needs to be remembered that there is a unique program such as the equake which has the high difference between the peak temperature and the steady-state; the difference is almost close to twice. Unless we prepare to such a unique pattern, the circuit may be damaged by the instantly sudden rise of temperature. Even if this occurs, as can be seen in Figure 8, our method may alleviate the damage to some extent.

3.5 Leakage Power and Performance

Leakage power is closely related to temperature. Generally, it is known that leakage power is exponentially increased by temperature rise. It is predictable that the leakage power may be reduced since the steady-state temperature was decreased by our method in the register file. We used HotLeakage [14] tool for calculating the leakage power, and the results can be seen from Figure 9a and Figure 9b. In most cases except one, the significant reductions of the leakage power are achieved, and the power saving is up to 24% on the average 13%. As described on the

Applied Computing Review 47
Section 3.3, since vortex showed the negative result from the temperature rise, the leakage power grew up at the rate of the temperature increase.

In addition, we can consider the benefit of performance improvement; as our logic eliminated several hotspots, the number of DTM-intervention will be reduced. The performance evaluation was carried out in limited four cases exceeding the emergency point: bzip2, crafty, gcc, and gzip. Because none of the other programs exceeded the emergency point, any performance drop will not exist; thus, it may be enough to analyze the only four cases. The emergency temperature was assumed to 80°C, and the used DTM technique reducing the temperature was the fetch-throttling. The fetch-throttling is a DTM scheme that suspends the instruction fetch if the temperature exceeds the emergency point until the temperature goes down to a safe area below the emergency point while the temperature is monitored. During the fetch stop, access counts to the register file will be reduced considerably; hence, the power density of the register file will be lowered, and the lowered power-density will lead to the temperature drop. However, such a DTM technique stopping the instruction pipeline decreases the Instruction Per Cycle (IPC) severely; therefore, the execution time for the given instructions will be significantly extended. Due to the extended time, the processor performance suffers severe damage. On the other hands, since the temperature of the register file is reduced by our architectural approach, the case beyond the critical temperature is extremely rare even in the most execution time. In other words, our method brings about the blocking effect; it fundamentally blocks a bad luck due to the performance loss by DTM. Figure 9c and 9d reports such a contrast result between the existing method and the proposed method, and Figure 9d presents the degree of the relative performance improvement due to the difference with Figure 9c.

Figure 10 shows the comparison of temperature changes over time under DTM-intervention between the existing method and our method. In other words, it shows the monitored results of the
temperature changes and performance loss by DTM; when the temperature exceeds the emergency point, DTM reduces the temperature of the register file by stopping the instruction fetch or by lowering the clock frequency. Figure 10a shows the temperature changes and the performance loss through the existing method, and Figure 10b shows the degrees of the temperature changes and the performance loss in the case of adopting our register-relocation method. First, In Figure 10a, the overall heating pattern is shown as the imbalanced form of the clustered access on the right side of the register-file as shown in the previous examples; thus, the maximum temperature is higher than Figure 10b. Also the initial temperature is formed highly as 87°C; accordingly, the DTM-intervention is immediately started due to the initial temperature exceeding the emergency temperature. The temperature of the register file starts to fall by stopping the instruction-fetch. It can be found that the temperature decreases below 80°C after 10 (x 100k) cycles as it rapidly goes down until about 10 (x 100k) cycles. Since the temperature dropped below the emergency point, the instruction-fetch is normally started again; however, the temperature of the register file will soon increase above 80°C by the restarted instruction-fetch. Then, DTM-intervention is triggered once again, the temperature falls again to below 80°C. As such a process continuously is repeated, the execution cycles of the instructions go on. Thus, from 10 (x 100k) cycles to 80 (x 100k) cycles in Figure 10a, it can be seen that the pattern like a saw blade is repeated. In other words the temperature is going up and down as the zigzag pattern around 80°C by DTM. Thus the execution time is increasingly extended; consequently, in Figure 10a, the execution time is spent about 110 (x 100k) cycles for finishing the given instructions.

Now let us look at the results of our relocation method in Figure 10b. As expected, it can be seen that the initial temperature is lower than Figure 10a; however, since there are some cases of exceeding 80°C, DTM is triggered at each time of the exceeding. Thus some patterns of the saw blades can be seen until about 20 (x 100k) cycles. But it is confirmed that because the saw blades patterns around 80°C are hardly seen after 20 (x 100k) cycles, there is little DTM-intervention compared to Figure 10a; therefore, the relative performance improvements can be expected compared to Figure 10a. As expected, this is due to the reduction of the maximum temperature since the overall temperature is distributed uniformly by our relocation technique. Consequently, the finished execution cycles are about 85 (x 100k) cycles, and the finished time is shorter than Figure 10a; accordingly, the performance of the processor will be relatively improved.

3.6 Overheads

Since our proposed logic consumes some power itself, it is necessary to consider the costs of our logic. The power model of our logic referred to that of CACTI [15] and Watch [8], and it was scaled to the technology of 65 nm. As the eight-bit adder and shift register consumes a big portion of the power, the attached logic increases the power consumption of the original renaming unit to 25%. However, since the power consumption of the existing renaming logic is relatively much smaller than the register file, the additional power consuming of our logic occupies only a trivial portion in the whole power consumption; the overall power consumption includes the consumption of the register file and that of the novel renaming unit.

Similarly, another overhead has to be considered in terms of temperature. As the area of a renaming unit is wider than a register file and the power consumption is quite smaller than the register file, the power density of the novel renaming unit is much lower than the register file; thus, it is expected that the temperature rise will be truly limited. From Table 2, it can be seen that the additional power consumption and the temperature increase are extremely slight.

Moreover, as it is considered that the reduction of leakage power and the performance improvement can be achieved by our logic, the relative overheads can be smaller.

<table>
<thead>
<tr>
<th>Table 2 Attached logic overheads</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overheads</strong></td>
</tr>
<tr>
<td>Power</td>
</tr>
<tr>
<td>(register file + renaming unit)</td>
</tr>
<tr>
<td>Temperature (renaming unit)</td>
</tr>
</tbody>
</table>
4. RELATED WORKS

Activity Migration [4] moves operations to multiple clones of register file when the emergency temperature is occurred. Thus the temperature of the previous unit can be decreased. But the migration overheads and area overheads exist and the migration overheads may damage the performance since the register file is time-critical. Active Bank Switching [2] proposed that only a few registers are active enough in the CPU cycles because not all the registers are used during the execution time. X.Zhou, et al. proposed the Compiler-driven Register Reassignment [3]. They evenly redistribute clustered registers to reduce the power density and temperature. But their method is the static one based on compiler and they only focused on the architectural registers. As can be seen until now, full-fledged research about the thermal-aware register file is not much yet.

5. CONCLUSIONS

The power density is rapidly being increased from high-performance microprocessors for servers to low-power embedded processors for mobile devices. Accordingly, generated heat and high temperature produce many thermal problems: malfunction, aging, leakage power, and cooling costs as well as the performance loss. The register file is the hottest unit of a microprocessor, and we have proposed a new renaming method for reducing temperature in the register file. Our idea can be implemented by adding a small logic to the traditional renaming unit and showed some results with around 1.5% overheads both power and temperature.

Our proposed method evenly redistributes allocation and access of the register file; ultimately, it lowers the overall power density for reducing the temperatures of the register file. As a result, we led to the temperature reductions reaching up to 10.4°C (11%) on average 4.6°C (6%); accordingly, hotspots were removed in some benchmark programs. The removed hotspots significantly reduces the performance loss by DTM-triggering; consequently, relative performance improvements reached up to 31.4% on average 24.8% compared to the traditional method. In addition, since leakage power is proportional to the exponential function of temperature, some leakage power reductions were obtained due to the temperature drop. Leakage power savings reached up to 24% on average 13%.

The dynamic renaming techniques proposed by us have produced some results with low overheads. Moreover, our logic may be able to cope with malicious attacks (e.g. thermal virus [12]) which may occur in the future because our logic has many chances to enhance the functionality by adding the appropriate response techniques in accordance with each application. For future work, we will continue to conduct research in this area.

6. ACKNOWLEDGEMENTS

This work was supported by the GRRC program of Gyeonggi province. [GRRC SUWON2010-B1), CCTV Image Based Context-Aware Process Technology]

7. REFERENCES

About the authors:

Jungwook Kim is a Ph.D. Candidate in the School of Electrical Engineering and Computer Science at Seoul National University in Korea. His research interests include thermal-aware architecture, power-density minimization, and low-power computer architecture. He received his B.S. (2003) degree in Rural Systems Engineering from Seoul National University in Korea.

Seong Tae Jhang is an associate professor in the Department of Computer Science at the University of Suwon in Korea. His research interests include multi-processor systems, computer architecture, and memory models. He received his B.S. (1986) degree in Computer Engineering from Seoul National University, his M.S. (1988) degree in Computer Engineering from Seoul National University, and his Ph.D. (1994) degree in Computer Engineering from Seoul National University.

Dr. Moon is a member of KISS, KICS, KIPS and KIIS. His research interests include real-time embedded systems, real-time databases, and real-time location-based tracking systems. He received his B.S. Degree from the University of Texas at Austin, 1986. He received his M.S. And Ph.D. Degrees in Computer Science from Florida State University in 1991 and 1997, respectively.

Chu Shik Jhon is a professor in the Department of Electrical Engineering and Computer Science at Seoul National University. He received his B.S. Degree in Applied Mathematics from Seoul National University in 1975, his M.S. Degree in Computer Engineering from Korea Advanced Institute of Science and Technology at Taegon in 1977, and his Ph.D. Degree in Computer Engineering from the University of Utah in 1982.