Frontmatter

2 Editors
3 S. Shin SIGAPP Semi-Annual Report
5 S. Shin Message from the Editor
6 H. Haddad SAC 2011 Preview

Invited Paper

8 An Efficient Fault Detection Algorithm for NAND Flash
    T. Kuo  P. Huang  Y. Chang
    C. Ko  C. Hsueh

Research Articles

17 Composite Ontology Matching with Uncertain Mappings Recovery
    C. d'Amato  N. Fanizzi
    F. Esposito

30 A Robust Link-Translating Proxy Server Mirroring the Whole Web
    Z. Mao  C. Herley

43 Coordinating e-Health Systems with TuCSoN Semantic Tuple Centres
    E. Nardini  A. Omicini
    M. Virolli  M. I. Schumacher
SIGAPP FY’11 Semi-Annual Report
July 2010- February 2011
Sung Y. Shin

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 Y. 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 to SIG Newsletters

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

SIGAPP's main event for this year will be the Symposium on Applied Computing (SAC) 2011 in Taichung, Taiwan from March 21-24 which will carry the tradition from Switzerland's SAC 2010. This year's SAC preparation has been very successful. More details about incoming SAC 2011 will follow in the next section. SIGAPP also supported several additional conferences with in-cooperation status and will continue partially sponsoring conferences, including ICUIMC and RACs this year.

We would also like to express our satisfaction with the new SIGAPP logo, best of a logo competition early last year, as it is now in full use for the SIGAPP web page and official publications such as the proceedings of SAC and Applied Computing Review.
A Message from the Editor

I’m pleased to announce the issuing of the winter issue of ACR (Applied Computing Review). As I promised in the previous issue, we will publish it semi-annually as an electronic version. However, once the format has stabilized, we will begin publishing quarterly electronically and in print.

I still have a dream that ACR will appear in the SCI (Science Citation Index). ACR contains invited papers from world-renowned researchers and selected papers presented by prominent researchers and professionals of 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 plan to invite some papers from ICUIMC and RACs conference, which are partially sponsored by SIGAPP.

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. I thank the authors for truly contributing to the state of the art in applied computing. 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.

We would like to express our sincere appreciation to the highly qualified peer reviewers that have coordinated an outstanding lineup of technical papers. We also would like to express our appreciation to the track chairs who served as the editorial board of ACR and the SIGAPP officers that served as associate editors of ACR. We would like to thank Ms. Irene Frawley as she has helped us get organized to begin publishing. I especially wish to thank Mr. Dan Stanton, technical editor of ACR, for his work on the design and layout of this new ACR, an international journal, and for his encouragement and support to publish this special issue. We also wish to thank all authors and Dr. Tei Wei Kuo and his researchers for their significant contributions. Without their hard work, this second issue of ACR would not be possible.

Next Issue
The planned release for the next issue of ACR is August 2011.
SAC 2011 Preview

The 26th Annual edition of the ACM Symposium on Applied Computing (SAC) will be held in Taichung, Taiwan. SAC 2011 is hosted by Tunghai University and sponsored by the Taiwan Ministry of Education, Taiwan Bureau of Foreign Trade, and Taiwan National Science Council. The local organizing committee consists of Dr. William Chu, Conference Chair; Dr. W. Eric Wong, Conference Vice Chair; Dr. Chih-Hsiong Shih, Tutorials Chair; and Drs. Chao-Tung Yang and Chih-Huang Chang, Local; Arrangement Chairs.

The conference will start on Monday with the Tutorials Program, organized by Dr. Chih-Hsiong Shih. The planned program is offering tutorials for all attendees free of charge. The local committee is planning a social luncheon for tutorial attendees who chose to participate, for a minimal fee. The program offers the following tutorials:

- Mitigation of Program Security Vulnerabilities: Approaches and Challenges
- Scenario-based requirements engineering facilitating interaction design
- Constraint Programming: From Algorithms to Applications
- Cloud Computing Security: From Fundamentals to Advanced Access Control
- Transition from Requirements to High-level Software Design

The Technical Program starts on Tuesday with the opening keynote address session by Dr. Philip S. Yu, from the University of Illinois at Chicago. Dr. Yu will talk about Mining Information Networks. In his talk, Dr. Yu will present various issues and solutions on scalable mining of information networks. These include data integration, data cleaning and data validation in information networks, summarization, OLAP and multidimensional analysis in information networks.

The three-day Technical Program, coordinated by Drs. Chih-Cheng Hung and Mathew Palakal, offers a wide range of sessions from the 40+ tracks organized this year by international groups of experts lead by Track Chairs and Co-Chairs. The complete listing of sessions and accepted papers in this year edition of SAC can be found on SAC 2011 website at http://www.acm.org/conferences/sac/sac2011/. This year, the program offers over 230 rigorously refereed papers. Accepted papers are organized into six themes: Applications, Distributed Systems, Engineering, AI and Agents, Information Systems, and Software Development.

This year, the open Call for Track Proposals has, after rigorous prescreening, accepted 40 new tracks for SAC 2011. The Call for Papers for these Tracks attracted 790 submissions from over 35 countries. The papers underwent a blind review process and resulted in the acceptance of 237 papers, making SAC 2011 acceptance rate just under 30%. In addition to the accepted papers, 59 papers that received high review scores were accepted as short papers for the Poster Program.

The Posters Program, coordinated by Dr. Jiman Hong, offers a selection of invited posters that represent high quality work from SAC Tracks. The posters program is planned for Wednesday, morning and afternoons sessions. A total of 59 posters will be presented.
This year, the organizing committee is planning two receptions on Monday and Tuesday. The Monday reception is sponsored by SIGAPP while the Tuesday reception is sponsored by SAC. In addition, free lunches are planned for Tuesday, Wednesday, and Thursday. Morning and afternoon coffee breaks are also planned for all four days of the conference. The conference Banquet event is scheduled for Wednesday evening at the Splendor Hotel. The Entertainment program and other social events have been planned by the local committee.

A paper from each theme will be selected as a Best Paper. A Best Poster will also be selected from the Posters Program. Winners will be honored during the Banquet event and will receive both certificates and prizes. Other awards will be presented during the Banquet event.

For those who plan to join us next month in Taichung and on behalf of the local host and organizing committee, we hope that you enjoy the program and have a pleasant stay at Tunghai University, which has one of the most beautiful college campuses in Taiwan. We wish you will have a great time at SAC 2011 and hope to see you at SAC 2012.

SAC 2012 Planning

The planning for the 27th edition of SAC is underway. SAC 2012 will be hosted by the Center for Computational and Systems Biology (COSBI) - The Microsoft Research - at the University of Trento, Italy. Italy was selected by SAC Steering Committee from three proposals: Korea, Italy, and Portugal.

The local organizing committee is lead by Dr. Paola Lecca and Professor Mirtis Conci, from COSBI, The Microsoft Research – University of Trento. A member of SAC Steering Committee, Dr. Sascha Ossowski, from the University Rey Juan Carlos, Madrid, Spain, will serve as the Conference Chair. The complete organizing committee can be found on SAC 2012 website at http://www.acm.org/conferences/sac/sac2012/. Local sponsors include Provincia Autonoma Di Trento and Riva del Garda Congressi. The conference dates are set for March 25 – 29, 2012. We hope you consider SAC 2012 for your next submission and hope to see you there next year.

On Behalf of SAC Steering Committee,

Hisham Haddad
Member of the Steering Committee
SAC 2011 and 2012 Treasurer and Webmaster
An Efficient Fault Detection Algorithm for NAND Flash Memory

Tei-Wei Kuo
Department of Computer Science and Information Engineering
Graduate Institute of Networking and Multimedia
National Taiwan University
Taipei 106, Taiwan, R.O.C.
ktw@csie.ntu.edu.tw

Po-Chun Huang
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan, R.O.C.
f95070@csie.ntu.edu.tw

Yuan-Hao Chang
Department of Electronic Engineering
National Taipei University of Technology
Taipei 106, Taiwan, R.O.C.
johnsonchang@ntut.edu.tw

Chia-Ling Ko
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan, R.O.C.
r96029@csie.ntu.edu.tw

Chih-Wen Hsueh
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan, R.O.C.
cwhsueh@csie.ntu.edu.tw

ABSTRACT
As flash memory gains its momentum in the storage market of embedded systems, existing fault detection algorithms face serious challenges due to the special characteristics of NAND flash memory and the rapid degradation of its reliability. This research proposes an efficient fault detection algorithm to detect the faults of NAND flash memory in a systematic way. Through the analysis of the testing time, the efficiency of the proposed algorithm is also evaluated and proved to be feasible.

Categories and Subject Descriptors

General Terms
Algorithms, Design, Management, Measurement, Performance, Verification

Keywords
flash memory, NAND, fault detection, march algorithm

1. INTRODUCTION
Proper fault detection capability is essential to the success of semiconductor-based chips, such as random access memory (RAM) and flash memory, especially when the capacity of memory chips has been growing nearly exponentially in the past years. As NAND flash memory\(^1\) is widely adopted in embedded systems as a storage medium, its unique access features and write-once property make many existing fault detection algorithms lack of efficiency or even infeasible, where the write-once property does not allow the rewriting of a page unless it is erased. Such an observation motivates this work in the proposing of a fault detection algorithm that is scalable and effective to detect potential faults with the considerations of the constraints of NAND flash memory.

The faults of NAND flash memory could be classified as two major types: Retention/endurance faults and initial faults. Retention/endurance faults could often be detected easily and corrected by error correction codes. Initial faults could be further partitioned into RAM-style faults and disturb faults, where initial faults should be detected before the factory shipment. Initial faults need efficient fault detection algorithms so as to reduce the fault detection cost. RAM-style faults, that are as the same as those of RAM, includes stuck-at-faults (SAF), transition faults (TF), stuck-open faults (SOF), and address decoder faults (AF) \(^1\). Disturb faults only occur on flash memory and are due to bit-line/word-line disturbances on access operations, where bit lines and

\(^{1}\)NAND flash memory has two major types: Single-level-cell (SLC) and multi-level-cell (MLC). Each cell of SLC/MLC\(_n\) NAND flash memory could contain \(1/(n)\)-bit information. In this paper, we focus on the fault detection of SLC NAND flash memory.
Cells get stuck at 0 or 1

I/O

1.5ms/block

2GB

Cells fail to transit from 1(0) to 0

I/O

200

(2K+64) Bytes

1

Cells and addresses are not an 1-to-

Cells are not accessible

1M

64

16K

0

54x-2417

2.1 Flash-Memory Faults

2. FAULTS AND DETECTION

In Section 3, fault detection algorithms for NAND flash memory are proposed. Section 4 analyzes the performance of the proposed algorithms. Section 5 is the conclusion.

This work is motivated by the needs of fault detection algorithms for NAND flash memory. An algorithm called NAND-march is proposed to detect all of the initial faults of SLC NAND flash memory in a systematic way. We also consider the effectiveness of the proposed testing patterns. With the performance analysis based on generic SLC NAND flash memory configurations, the efficiency and feasibility of the proposed algorithms are validated.

This paper is organized as follows: Section 2 summarizes fault models and existing popular fault detection algorithms. In Section 3, fault detection algorithms for NAND flash memory are proposed. Section 4 analyzes the performance of the proposed algorithms. Section 5 is the conclusion.

2. FAULTS AND DETECTION

2.1 Flash-Memory Faults

A NAND flash memory chip is partitioned into blocks, and each block is further divided into a fixed number of pages. A block and a page are the units of erase and read/write operations, respectively. A written page can not be overwritten unless its residing block is erased. A page consists of flash cells. When each cell stores 1-bit information, the flash memory chip is called a single-level-cell (SLC) flash memory chip. Each cell of a multiple-level-cell \(x\times_n\) (MLC\(x\times_n\)) flash memory chip can store \(n\)-bit information. Cells of a block are interconnected through word lines and bit lines, as shown in Figure 1. A word line (referred to as a row) usually connects the cells of a page together, and a bit line (referred to as a column) usually connects cells that have the same offset to their corresponding page (referred to as a string). A write/program operation can change the status of a cell from 1 to 0, where an erase operation can change the status of a cell from 0 back to 1. In other words, a write operation can change the status of cells with electrons, where only the cells that are changed from 1 to 0 are charged. An erase operation would remove electrons from every cell of a block. As a result, the time to erase a block is much longer than that to write/program a page, as summarized in Table 1 [16].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Size</td>
<td>(2K+64) Bytes</td>
</tr>
<tr>
<td>Pages per Block</td>
<td>64</td>
</tr>
<tr>
<td>Blocks per Chip</td>
<td>16K</td>
</tr>
<tr>
<td>Pages per Chip</td>
<td>1M</td>
</tr>
<tr>
<td>Chip Capacity</td>
<td>2GB</td>
</tr>
<tr>
<td>Serial Access</td>
<td>25ms/byte</td>
</tr>
<tr>
<td>Random Read</td>
<td>20µs/page</td>
</tr>
<tr>
<td>Write/Program</td>
<td>200µs/page</td>
</tr>
<tr>
<td>Erase</td>
<td>1.5ms/block</td>
</tr>
</tbody>
</table>

Table 1: Typical Configurations for a Generic SLC NAND Flash Memory Chip

Faults of flash-memory cells can be classified as retention (or endurance) faults and initial faults [17]. Retention/endurance faults occur after cells are repetitively written and erased, and they can be detected by the error correction codes (ECC). Initial faults are mainly due to defects before the factory shipment, and they often become worse as the density of flash-memory chips increases. Initial faults are usually resolved by some fault detection processes so that flash-memory chips with excessive initial faults are discarded. Unfortunately, many existing fault detection processes are either very time-consuming or lack of the capability to detect initial faults effectively.

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{RAM-Style Faults} & \text{Stuck-At Fault (SAF)} & \text{Cells get stuck at 0 or 1} \\
\text{Transition Fault (TF)} & \text{Cells fail to transit from 1(0) to 0(1)} \\
\text{Stuck-Open Fault (SOF)} & \text{Cells are not accessible} \\
\text{Address Decoder Fault (AF)} & \text{Cells and addresses are not an 1-to-1 mapping} \\
\hline
\end{array}
\]

<table>
<thead>
<tr>
<th>Disturb Faults</th>
<th>(0 \rightarrow 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word-Line Erase Disturb (WED)</td>
<td>(0 \rightarrow 1)</td>
</tr>
<tr>
<td>Word-Line Program Disturb (WPD)</td>
<td>(1 \rightarrow 0)</td>
</tr>
<tr>
<td>Bit-Line Erase Disturb (BED)</td>
<td>(0 \rightarrow 1)</td>
</tr>
<tr>
<td>Bit-Line Program Disturb (BPD)</td>
<td>(1 \rightarrow 0)</td>
</tr>
</tbody>
</table>

Table 2: Initial Faults: RAM-Style Faults and Disturb Faults

Initial faults of flash memory can be further classified as RAM-style faults and disturb faults (DF) [17], as shown in
Figure 2: Address Decoder Faults

Table 2. *RAM-style faults* are the ones that occur at individual cells and might happen to both RAM and NOR/NAND flash memory. Example RAM-style faults include Stuck-At Fault (SAF), Transition Fault (TF), Stuck-Open Fault (SOF), and Address decoder Fault (AF). The value of a cell with SAF would get stuck at 0 or 1. A cell with TF can not change its value from 0/1 to 1/0. With SOF, a cell can not be accessed due to the faults on its corresponding lines. As a result, the retrieved value would be the one that was input to the sense amplifier. AF refers to a non-one-to-one mapping between cells and their addresses, and there are four types, i.e., $AF_{na}$, $AF_{nc}$, $AF_{ma}$, and $AF_{mc}$, as shown in Figure 2. $AF_{na}$/$AF_{nc}$ refers to a fault in which an address(/cell) is not connected to any cell(/address). On the other hand, $AF_{ma}$/$AF_{mc}$ refers to a fault in which an address(/cell) is connected to more than one cell(/address). Different from RAM-style faults, *disturb faults* only happen to flash memory. Defect cells could be disturbed when other cells of the same word line or bit line are being accessed. As shown in Table 2, there are four types of disturb faults: WED(WPD) results in unexpected erasing(/programming) of cells when other cells of the same word line are being accessed. BED(BPD) results in unexpected erasing(/programming) of cells when other cells of the same bit line are being accessed.

### 2.2 Fault Detection

<table>
<thead>
<tr>
<th>March Expression</th>
<th>Detection Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>March Test Module</td>
<td>[March Element; March Element; ...]</td>
</tr>
<tr>
<td>March Element $T_e$ (Operation, Operation, ...)</td>
<td></td>
</tr>
<tr>
<td>$\uparrow$ Ascending address order</td>
<td></td>
</tr>
<tr>
<td>$\downarrow$ Descending address order</td>
<td></td>
</tr>
<tr>
<td>$\uparrow\downarrow$ Either asc. or desc. address order (or arbitrary order)</td>
<td></td>
</tr>
<tr>
<td>Operation $r$.x</td>
<td>Read data and expect the value to be $x \in {0, 1}$</td>
</tr>
<tr>
<td>w.x</td>
<td>Write a value $x \in {0, 1}$; w:1 is forbidden in flash</td>
</tr>
<tr>
<td>e</td>
<td>Erase a block, changing all cells to value 1</td>
</tr>
</tbody>
</table>

Table 3: Symbol Table for March Algorithms

In the past years, excellent fault detection algorithms are proposed to detect initial faults of byte-addressable storage media such as RAM and NOR flash memory [2, 3, 4, 18, 19, 20, 21]. In particular, the *March algorithm* could detect RAM-style faults and also the disturb faults of NOR flash memory [2, 3, 21]. A *march test module*, that is an implementation of a march algorithm $M_e$, is composed of $m$ march elements $\{S_1; S_2; \ldots; S_m\}$ to detect some specific type $c$ of faults. A march element consists of a sequence of read, write, or erase operations in an ascending or descending address order, as shown in Table 3. For example, a march element “$r_1, w_0, r_0$” issues three operations to each cell in the ascending (cell) address order. That is, the march element expects to read 1 from a cell first. Writing 0 to the cell should be then followed by a reading of 0 from the cell. The march element moves over cells in an ascending (cell) address order. A fault is detected if a returned value is not the expected one indicated by the read operation.

Table 4: Testing Modules

Table 4 summarizes existing march-like fault detection algorithms with minor modifications to satisfy the write-once property of NOR flash memory [22, 2, 3, 13, 15]. The $M_{saf}$ (referred to as the MSCAN method or Zero-One algorithm) is to detect stuck-at-0 faults ($SAF_0$) and then stuck-at-1 faults ($SAF_1$). The $M_{tf}$ further ensures that cells can also transit from 0 to 1 so as to detect any transition fault. The $M_{dof}$ is to detect stuck-open faults by ensuring that the previous returned value is not latched in the output of the sense amplifier. The $M_{df}$ detects the address decoder faults in both directions (i.e., ascending and descending address orders) because an address might be connected to multiple cells, as shown in Figure 2(d). Different from previous ones, the $M_{df}$ is to detect whether a write operation to a cell disturbs the value of another cell, and this testing process is proceeded in both directions so that the disturb faults on edge cells can be detected.

Although march-like algorithms could detect RAM-style faults and disturb faults of NOR flash memory, they are too time-consuming to be practical in field tests because of the rapidly growing capacity of flash memory chips. Even their efficient variations, e.g., March-FT [3], might still face severe challenges. On the other hand, march-like algorithms could not detect every initial fault of NAND flash memory because NAND flash memory is not byte-addressable, and pages can not be overwritten unless their residing blocks are erased. As a result, the industry usually adopts some heuristics in fault detection. A popular one is to write all-0 and/or all-1 patterns to each page and then read every one of them to verify whether the written values are stored and retrieved correctly. Another well-known detection algorithm is to write so-called checkerboard patterns and then read the corresponding pages to verify the correctness of cells, where a checkerboard pattern interleaves 0’s and 1’s, such as a bit pattern “0101”. We must point out that such heuristics-based approaches could only detect few types of faults and do not provide a systematic and/or scalable way for fault detection. Such an
3. FAULT DETECTION ALGORITHMS FOR NAND FLASH MEMORY

3.1 Overview

This section is meant to propose fault detection algorithms for NAND flash memory, since existing march-like fault detection algorithms can not be directly applied to the fault detection of NAND flash memory. The setting of one page as an access unit (instead of one bit) would simply introduces problems in the detection of certain fault types because of the large differences in physical characteristics between NOR and NAND flash memory.

In NAND flash memory, the definition of the conventional march element is no longer suitable because the testing of a NAND flash-memory chip should consider the testing order of blocks in the chip as well as the testing order of pages in each block. On the other hands, the conventional march algorithms could not detect the bit-line address decoder fault and the word-line disturb fault. The bit-line address decoder fault is caused by that some bit-line addresses do not correctly correspond to their respective bit-lines, and the word-line disturb fault occurs because defect cells are disturbed (or unexpectedly erased or programmed) when other cells of the same word line are being accessed. Because data of a page are written by the same write operation, the bit-line address decoder fault and word-line disturb fault could not be identified by writing the conventional all-0(/1) patterns and could not be efficiently detected by the diagonal-0(/1) patterns. Even when the march algorithm $M_{af}$ for address decoder faults and the march algorithm $M_{df}$ for disturb faults are adopted, the bit-line address decoder fault and the word-line disturb fault still could not be detected because $M_{af}$ and $M_{df}$ assumes that cells at different bit lines of the same word-line could be programmed individually.

In order to detect the above two faults for NAND flash memory, special bit patterns (referred to as NAND-march pattern) are designed on the basis of a page (see Section 3.2). By adopting the NAND-march pattern, a NAND-march algorithm with the considerations of the testing order of pages in each block (see Section 3.3) is proposed to detect the RAM-style faults and disturb faults, including the bit-line address decoder fault and word-line disturb fault that could not be detected by conventional march-like algorithms.

In the proposed NAND-march algorithm, the symbol for a march element is revised as $T_{po}(\text{Operation}, \text{Operation}, \ldots)$, where $T_{po}$ represents the page testing order, i.e., $⇑$, $⇓$, or $⇔$; and Operation represents the operation to read/write a page (i.e., $rP/wP$) or erase a block (i.e., $e$) (see Table 5). Note that $P$ is the testing bit pattern for a flash page, and $\bar{P}$ is the bit-wise complement of $P$.

3.2 NAND-March Pattern

To detect the bit-line decoder faults, the diagonal bit patterns could be easily extended for NAND flash memory. As shown in Figure 3(a), suppose that each block contains 8 pages, and each page consists of 10 bits. The first time to write the diagonal-0 pattern could not detect whether writing data to the last two bit lines would affect data of other bit lines due to the bit-line address decoder faults. As a result, the whole block needs to be erased for the second run with the diagonal-0 pattern written to the last two bit lines. After that, the diagonal-0 pattern should be bit-widely inverted to detect the stuck-at fault, transition fault, and stuck-open fault, as shown in Figure 3(b). In practice, in order to finish the diagonal pattern, it needs at least $2\times\lceil \frac{N_p}{N_b} \rceil$ test runs (or block erases), where $N_b$ is the number of bits in a page and $N_p$ is the number of pages in a block. When the page size keeps increasing, the ratio of $N_b$ to $N_p$ grows to worsen the performance of the detection algorithms that adopt the diagonal pattern. As a result, the diagonal pattern is too time-consuming to be practical. For example, given a flash chip with 128-page blocks and 4KB pages, the number of test runs (or block erases) would be up to 512 (i.e., $2 \times \lceil \frac{496 \times 8}{128} \rceil$).

In this section, a NAND-march pattern is proposed to improve the efficiency on the bit-line decoder fault detection. The proposed NAND-march pattern includes a parallel pattern and an inverse parallel pattern, as shown in Figures 4(a).
3.3 NAND-March Algorithm

By adopting the NAND-march pattern, a NAND-march algorithm is proposed to detect faults with the considerations of the NAND flash memory characteristics. Equation 1 shows the algorithm of the proposed NAND-march algorithm, where \( \mathcal{P}(r/w) \) expects to read data from a page with \( 1's/(0's) \) and \( w(1/w) \) represents to write a page with all \( 1's/(0's) \). Note that \( \mathcal{P}_{NAND} \) represents the bit pattern for a page and is changed according the proposed NAND-march pattern (see Figure 4).

\[
\mathcal{P}_{NAND} = \{ \mathcal{P}(r/w), \mathcal{P}(r/w) \} \quad 1 \leq r/W \leq w \leq W
\]

This algorithm consists of nine march elements that are partitioned into three test runs, each of which is composed of three march elements and starts with an erase operation to erase the whole flash chip. In the first run, the NAND-march algorithm erases the whole flash chip. The second march element expects to read 1 from each cell, and then writes 0 to change the status of each cell, followed by a read operation with expecting all-0 values, so as to detect the stuck-at fault. Meanwhile, the stuck-open fault could be also detected in the second march element because its second read operation to a page that has some stuck-open-fault bits. After that, the third march element expects reading 1 from each page cell so as to make sure that the word-line erase disturb doesn’t occur. The second run starts with resetting the status of each cell back to 1. The fifth march element expects to read 1’s from a page, writes the \( \mathcal{P}_{NAND} \) pattern to a page, and expects to read the \( \mathcal{P}_{NAND} \) pattern from the page in the ascending page address order. Note that the \( \mathcal{P}_{NAND} \) pattern is designed to help on the detection of the transition fault, address decoder fault, and disturb faults (see Section 3.2). Until the fifth march element, each cell has already flipped from 1 to 0 and turned back to 1 if it doesn’t have the transition fault. Due to the bit-line disturb fault consideration, the sixth march element is performed to read each page to make sure that cell values are not disturbed. Finally, the third test run is to perform with the inverse parallel pattern, so as to complete the detection of the word-line disturb fault.

4. PERFORMANCE EVALUATION

The purpose of this section is to evaluate the capability of the proposed fault detection algorithm (i.e., the NAND-march algorithm) in terms of testing performance. In this evaluation, a 2GB NAND flash memory is evaluated (see Table 1) without the adoption of any hardware acceleration operations, such as the two-plane operation and the copy-back operation [16]. Well-known march algorithms are evaluated to compare the efficiency and feasibility of the proposed algorithm, even though they could not detect some types of faults of NAND flash memory.
Table 6: Testing Time and Targeting Fault Types of Each Module of NAND-March Algorithm

<table>
<thead>
<tr>
<th>Module</th>
<th>Time (sec.)</th>
<th>Acc. Time (sec.)</th>
<th>SAF</th>
<th>Sof</th>
<th>TF</th>
<th>AF</th>
<th>DF</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₁: (e)</td>
<td>24.576</td>
<td>24.576</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>M₂: ⊣ (r₁, w₀, r₀)</td>
<td>417.753</td>
<td>442.329</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>M₃: ⊣ (r₀)</td>
<td>76.336</td>
<td>518.666</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>M₄: (e)</td>
<td>24.576</td>
<td>543.241</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>M₅: ⊣ (r₁, wPₙₐₙ₃₋₁, rPₙₐₙ₃₋₁)</td>
<td>417.753</td>
<td>960.994</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>M₆: ⊣ (rPₙₐₙ₃)</td>
<td>76.336</td>
<td>1037.330</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>M₇: (e)</td>
<td>24.576</td>
<td>1061.906</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>M₈: ⊣ (r₁, wPₙₐₙ₃₋₁, rPₙₐₙ₃₋₁)</td>
<td>417.753</td>
<td>1479.659</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>M₉: ⊣ (rPₙₐₙ₃)</td>
<td>76.336</td>
<td>1555.995</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 7: Performance Comparison among NAND-March and Existing March Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND-march</td>
<td>1555.995</td>
</tr>
<tr>
<td>M₉ₐ₉</td>
<td>442.329</td>
</tr>
<tr>
<td>M₉</td>
<td>543.241</td>
</tr>
<tr>
<td>M₉₀</td>
<td>442.329</td>
</tr>
<tr>
<td>M₉₁</td>
<td>884.658</td>
</tr>
<tr>
<td>M₉ for NOR flash</td>
<td>884.658</td>
</tr>
<tr>
<td>Sum of Existing Work</td>
<td>3197.213</td>
</tr>
</tbody>
</table>

6. REFERENCES


[22] Mohammad Gh. Mohammad, Kewal K. Saluja, and

About the authors:

Tei-Wei Kuo received his B.S.E. and Ph.D. degrees in computer science from the National Taiwan University and the University of Texas at Austin in 1986 and 1994, respectively. He is a Distinguished Professor of the Department of Computer Science and Information Engineering, National Taiwan University, where he served as the Chairman from August 2005 to July 2008. Prof. Kuo is an IEEE Fellow. He serves in the editorial board of the Journal of Real-Time Systems and IEEE Transactions on Industrial Informatics. He was a Program Chair and a General Chair of the IEEE Real-Time Systems Symposium and now chairs the Embedded Systems Group of the Networked Communication Program Office of Taiwan and serves as an independent board director of the Genesys Logic and the MStar Semiconductor. His expertise is on real-time systems, embedded systems, and storage systems.

Po-Chun Huang received his B.S. degree in Computer Science from National Taiwan University, Taipei, Taiwan (R.O.C.) in 2005. He is currently a Ph.D. candidate of the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan (R.O.C.). His research interests include flash memory, storage systems and operating systems.

Yuan-Hao Chang received his B.S. degree in Information and Computer Education from National Taiwan Normal University, Taipei, Taiwan, in 1997. He received his M.S. degree in Computer Science and Information Engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2004. He received his Ph.D. degree in Networking and Multimedia of Computer Science and Information Engineering from National Taiwan University, Taipei, Taiwan, in 2009. He is an IEEE member. Now he is an assistant professor at the Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan, since Feb. 2010. Previously, he was a software/firmware design engineer in VeriFone, a division of Hewlett-Packard, Taipei, Taiwan, between Sep. 2000 and Sep. 2002, where he joined OMNI series POS (Point of Sales) system development projects to implement drivers for devices such as card readers, thermal printers, and internal modems/Ethernets. His expertise is on real-time systems, embedded systems, operating systems, and storage systems.
Chia-Ling Ko received his B.S. degree in Computer Science from National Chung-Cheng University, Chayi, Taiwan (R.O.C.) in 2007. She received her M.S. degree in Computer Science from National Taiwan University. Now she is a software engineer in MediaTek Inc.

Chih-Wen Hsueh received the B.E. degree from the National Taiwan University, Taipei, R.O.C., in 1989 and the Ph.D. degree in information and computer science from the University of California at Irvine, CA, USA, in 1997.

Since 2006, he has been with the Graduate Institute of Networking and Multimedia and Department of Computer Science and Information Engineering at the National Taiwan University, R.O.C., where he is currently an Assistant Professor. In 1999, he was an Assistant Professor with the National Chung Cheng University, R.O.C., and, in 1999, he was an Associate Research Engineer with the Institute of Information Science at the Academia Sinica. His research interests include real-time systems, operating systems, embedded systems, and distributed systems.
An automated ontology matching methodology is presented, supported by various machine learning techniques, as implemented in the system MoTo. The methodology is two-tiered. On the first stage it uses a meta-learner to elicit certain mappings from those predicted by single matchers induced by a specific base-learner. Then, uncertain mappings are recovered passing through a validation process, followed by the aggregation of the individual predictions through linguistic quantifiers. Experiments on benchmark ontologies demonstrate the effectiveness of the methodology.

Categories and Subject Descriptors
D.2.12 [Interoperability]: Data mapping; I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods; H.3.1 [Content Analysis and Indexing]: Linguistic processing; I.2.6 [Artificial Intelligence]: Learning

Keywords
Ontology matching, Uncertainty, Validation, Aggregation

1. INTRODUCTION
The Semantic Web (SW) [3, 30] is the new vision of the Web whose goal is to make the Web content machine readable and processable besides of human-readable. This view is grounded on the availability of domain ontologies [18] to be used for semantically annotating resources.

An ontology typically provides a (computer format) vocabulary that describes a domain of interest and a specification of the meaning of terms used in the vocabulary. The standard language for ontological representation in the SW context is OWL [5] language, which is grounded on Description Logics\(^1\) (DLs) [2] as theoretical framework. However, in open and evolving systems, such as the SW, different parties adopt different ontologies. This raises heterogeneity problems at a higher level [13].

Ontology matching [13] is a promising direction towards the solution of the semantic heterogeneity problem. It focuses on finding correspondences between semantically related entities of different ontologies thus enabling the knowledge and data expressed in the matched ontologies to interoperate.

Although a variety of automatic ontology matching systems have been proposed so far, their performance may vary a lot depending on the different domains [9]. This problem is generally tackled by selecting the optimal matcher based on the nature of the matching task and the different features of the systems. This selection may involve Machine Learning techniques [26] for finding optimal configurations of the matchers, determining the appropriate heuristics / parameter values to achieve the best results [10].

We propose a comprehensive approach that differs from the previous ones for exploiting a combination of multiple matchers which are able to capture diverse aspects of the alignment. This should allow to overcome the weakness of the individual matchers. The idea of ensemble learning is inducing multiple classifiers (matchers) so that the accuracy of their combination (different classifiers can complement and complete one another) may lead to a higher performance.

The proposed methodology is made up of two stages. In the first stage it uses individual base-learners and then a meta-learner to elicit certain mappings from those predicted by single matchers induced by the base-learners. This phase adopts the stacking [36], an ensemble learning technique, which seems the most appropriate for composing diverse learners. In the second stage, mappings that were previously deemed as uncertain can be recovered through a taxonomic / structural validation process, followed by the aggregation of the individual predictions made by linguistic quantifiers [17]. This stage may be considered as a way of enriching the features utilized for the choice of the best matchers, in order to get a more effective combination [9].

The methodology is validated in a realistic setting on benchmark ontologies. In particular, we use datasets from past OAEI campaigns\(^2\) that provide a gold standard mapping as well as mappings created by different matching systems. Thus, our system can train a classifier on the outcome of

\(^1\)Description Logics constitute a fragment of First Order Logic characterized by a well defined formal semantics and a set of available deductive inference services.

\(^2\)http://oaei.ontologymatching.org
different matching systems and learn what combination of results from different matchers with the best indication of a correct correspondence. This is competitive w.r.t. previous attempts of combining matchers which have often been based on ad hoc methods or had to be customized manually.

The paper proposes the following contributions:

- a hybrid approach for combining various matching systems using machine learning techniques and linguistic aggregation;
- a methodology which is especially meant to recover uncertain mappings (also called candidate mappings in the rest of the paper):
  - through a number of validators adopting linguistic and structural similarity functions [29];
  - and an aggregation operator implementing a large choice of quantifiers [17];
- experiments on OAEI benchmark ontologies prove that recovering mappings (through validation and aggregation) can significantly improve the overall performance of the ontology matching system (especially in terms of recall).

The remainder of the paper is organized as follows. Firstly, the architecture of the MoTo system is presented in Sect. 2. Then in section 3, we illustrate the algorithms used to validate and aggregate multiple matchers. Experiments proving the effectiveness of the method are illustrated in Sect. 4. In Sect. 5 related works are examined. Finally, Sect. 6 concludes the work outlining possible developments.

2. AN ONTOLOGY MAPPING SYSTEM

Let $O = (N_C, N_R, N_I)$ denote the input ontology, where $N_C$, $N_R$ and $N_I$ stand, respectively, for the sets of the names for the concepts (classes), the roles (properties) and the individuals of the ontology. For simplicity, we will consider the problem of matching the concepts of some ontology $O_1 = (N^1_C, N^1_R, N^1_I)$ to those of another ontology $O_2 = (N^2_C, N^2_R, N^2_I)$. We will focus on the problem of finding equivalence mappings ($\equiv$), although the method could be extended to discover subsumption mappings ($\sqsubseteq$).

2.1 Application Context

The reference application context of our hybrid ontology system is illustrated by Fig. 1. Given two input ontologies under comparison $O_1$ and $O_2$, let us suppose that, in order to assess the semantic correspondences among the entities in the ontologies, the matching system eventually provides a similarity matrix for the entities belonging to either ontology. Namely each element of the matrix contains a value that indicates the similarity between the couple of entities related to the row and column. This may have been computed through any specific technique which is aimed at determining such a value. On the ground of this matrix, a decision making module will determine the certain mappings (those whose similarity value exceeds a given threshold $\varepsilon$) from the others. If the system discarded all other mappings, it would likely commit some errors, especially with those mappings that yielded a similarity which was not far from the threshold. Hence a number of uncertain mappings (i.e. a sub-matrix of the whole similarity matrix) are retained as candidate mappings to be evaluated along further techniques. To this purpose a taxonomic validator and a structural validator have been devised. They shall be discussed later in this section. Likewise, aggregation operators can be applied in the perspective of combining more validators. In such a perspective, the problem of assigning a degree of trust to each validator arises. In the following, we discuss our method for coping with this problem.

2.2 System Architecture

MoTo (Mapping ontology To ontology) is a multistrategy ontology mapping system. It resorts to a variety of matching techniques based on machine learning, linguistics and structural criteria. Additionally, MoTo also implements aggregation principles for the mappings in order to evaluate the nature of the uncertain mappings, namely if they have to be taken into account or not. Fig. 2 shows its architecture.

The system is among the composite systems, according to the classification in [31, 13]. The main functions implemented as separate modules are:

- instance classification. All individuals from an ontology are collected and the membership of each of them w.r.t. each concept of the ontology is determined along with both a deductive and an inductive approach. Specifically, first of all, by the use of a standard deductive reasoner (e.g. pellet reasoner), the instance checking for each individual in the ontology is computed with respect to each concept in the same ontology. Due to

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure1.png}
\caption{The reference application context.}
\end{figure}
the *Open World Assumption*, typically made in the SW context, there could be some unknown cases, namely individuals for which it is not possible to prove neither if they are instance of a given concept nor of the negated concept (used to assert that an individual is not instance of the concept). In order to manage these unknown cases, these individuals are inductively classified with respect to each concept in the ontology by the use of the k-Nearest neighbor algorithm adapted to the SW representation [14, 4] and the individuals are assigned to the concepts that are majority voted by the training examples that are most similar to the individuals to classify (the Tversky measure [34] is adopted as a similarity measure).

- **distribution estimation.** This module works on the ontologies to be matched utilizing the services of a separate ML module to estimate the probability distribution, related to each couple of concepts \((C_1, C_2) \in N_1 \times N_2\), that will be successively used for computing the similarity matrix. The ML module currently includes four *base-learners* and one *meta-learner* (implementing the technique of stacking of base-classifiers produced by base-learners) to combine their predictions [36]. The available base-learners produce a \(k\)-Nearest Neighbor classifier, an *Artificial Neural Network* and two *Bayesian Classifiers*, i.e. a *content* and a *name* classifier (in this case the information about possible superclasses is also exploited) [35].

This module is used for estimating, for each couple \((C_1, C_2)\), the joint probability of membership of the individuals to either concept. Each base-learner \(L\) receives a dataset made up of individuals from \(O_1\) divided in the group of members of \(C_1\) and the group of non-members of \(C_1\). This is exploited to train a base-classifier. Then the individuals in \(O_2\) are divided in two groups according to their membership to \(C_2 \in N_2\). The classifier trained on the instances in \(O_1\) is then used to classify the instances in \(O_2\). Thus, four instance sets are formed, \(U_1^{C_1, C_2}, U_1^{\neg C_1, C_2}, U_1^{C_1, \neg C_2}\) and \(U_1^{\neg C_1, \neg C_2}\), respectively corresponding to the individuals that belong to both \(C_1\) and \(C_2\), the individuals that do not belong to \(C_1\) but belong to \(C_2\) and vice-versa and the individuals that belong neither to \(C_1\) nor to \(C_2\).

The same procedure is repeated swapping the roles of the concepts (and related instances), namely by training each base-learner on the individuals that are instances and not instances of \(C_2\) and classifying the
individuals that are instances and not instances of \( C_1 \). Thus, the four instance sets \( U_1^{C_1 \cap C_2}, U_1^{C_1 \cap \neg C_2}, U_1^{\neg C_1 \cap C_2}, U_1^{\neg C_1 \cap \neg C_2} \) are obtained. The whole procedure can be iterated for each of the available base-learner. This is likely to lead to different outcomes, i.e. different \( U \)-sets.

A simple procedure to make a decision in controversial classification cases may be based on a majority vote, which proves a sub-optimal but often a quite effective method [9]. Alternatively, a more effective technique such as stacking [35] may come into service (for instance when the number of the base-learner is even and there is no majority among the classification results of the base-learner). It consists in adopting a meta-learner which produces the final decision on the ground of the determination of the base-learners. The results provided by the ML-module are passed to the distribution estimator which computes the joint distributions related to the membership / non-membership to the two concepts as follows: \( P(C_1, C_2) = \left( |U_1^{C_1 \cap C_2}| + |U_1^{C_1 \cap \neg C_2}| \right) / (|N_1^1| + |N_1^2|) \) where \(| \cdot |\) stands for the set cardinality. The other joint probability distributions (\( P(\neg C_1, C_2), P(C_1, \neg C_2), P(\neg C_1, \neg C_2) \)) can be derived analogously.

• similarity estimation. This module receives the joint probability distributions computed by the distribution estimation module and exploit them for determining the concept similarity. Specifically, the following similarity measures can be considered [23, 16]:

- Jaccard\((C_1, C_2) = a/(a+b+c)\)
- Dice\((C_1, C_2) = (b+c)/(2a+b+c)\)
- Ochiai\((C_1, C_2) = a/\sqrt{(a+b)(a+c)}\)
- GowerLegendre\((C_1, C_2) = (a/(b+c)+d)/(a+b+c+d)\)

where \( a = P(C_1, C_2), b = P(C_1, \neg C_2), c = P(\neg C_1, C_2) \) and \( d = P(\neg C_1, \neg C_2) \).

A similarity matrix \( S \in \mathbb{R}^{|N_1| \times |N_2|} \) is computed where each element \( s_{ij} \in [0, 1] \) represents the similarity of the couple of concepts \( (C_i, C_j) \in N_1 \times N_2 \).

Two thresholds \( (\theta_{\min}, \theta_{\max}) \) are determined to separate certain \( \{ (C_i, C_j) \in N_1 \times N_2 | s_{ij} \geq \theta_{\max} \} \), discarded \( \{ (C_i, C_j) \in N_1 \times N_2 | s_{ij} < \theta_{\min} \} \) and uncertain mappings \( \{ (C_i, C_j) \in N_1 \times N_2 | \theta_{\min} \leq s_{ij} \leq \theta_{\max} \} \). The first group can be output, while the second one is simply discarded, while the last one is subject to further computation (candidate mappings).

• validation. This module is responsible for the further evaluation of uncertain mappings. Specifically, the similarity of the concepts involved in uncertain mapping is re-computed by the use of different validators each one grounded of different criteria.

Two types of validators were developed: text-based validators that use the vocabulary in conjunction with the ontological model (it will not be further detailed in the following) and structure-based validators that use the taxonomic structure of the input ontologies or also the entire graph established by the relationships (properties) therein.

• aggregation. The aggregation operator is activated at the end of the whole process for the composition of the results provided by the different validators to give a single decision regarding the uncertain mappings.

3. VALIDATION AND AGGREGATION

3.1 Taxonomic and Structural Validation

In order to compute the similarity between concepts involved in uncertain mappings, different criteria from the ones used for calculating the similarity matrix have to be taken into account. Following [13], five criteria have been considered for the comparison of two concepts:

#1 most of their direct super-concepts are similar;
#2 most of their direct sub-concepts are similar;
#3 most of their super-concepts are similar;
#4 most of their sub-concepts are similar;
#5 all of their related concepts and properties are similar.

Criteria #1-#4 merely employ the taxonomic relationship between concepts in the ontology. Validation following these criteria can be determined by a taxonomic validator. The final criterion considers also other relationships determined by the properties in the ontologies. This is implemented in a structural validator.

3.1.1 Taxonomic Validation

The taxonomic validation module is based on the idea of comparing two concepts based on their respective position within the related subsumption hierarchy. The candidate mappings found to be uncertain in the previous phase are input to this validator to make a decision on their final rejection or a possible recovery. By analyzing the taxonomies (parenthood relationships between the concepts) the module re-computes a similarity value for the candidate couples of concepts along the criteria #1-#4 above.

The taxonomic validation module requires the initial similarity matrix \( S_i \) (sub-matrix of the similarity matrix \( S \)) containing the values that determined certain mappings and the couples of uncertain mappings that are to be validated. Each concept \( C_i \in O_i \) is assigned with a unique concept \( C_i \in O_2 \) so that \( S_i \) is transformed in a sparse matrix (since, for each row, only one value is different from zero).

Fig. 3 depicts the algorithm for the \( n \)-th criterion. This algorithm computes the similarity of two concepts on the ground of the similarity of their (direct) parenthood. Moreover, both the observed average similarity and the rate of matches found are taken into account. They are carefully combined since it should be considered that although the average similarity may be high the hierarchical structure of the ontologies should be also taken into account. This means that even in presence of a high similarity value on average, this value could be decreased if the structure of the parenthood of the two considered concepts is very different (for instance in the number of concepts that are included).

After computing the similarity of all couples of concepts from either ontology according to \( \text{SIM}_i(C_i, C_j), i = 1, \ldots, 4 \), the final similarity value given by the validator is a linear combination of these values with an optional additional term, which stands for a similarity value computed via another function \( \sigma \) covering a different aspect.
Given the set $RE(C_1, C_2)$ of the couple of related concepts w.r.t. $C_1$ and $C_2$, the structural similarity measure $\text{sim}_s : N^2 \times N^2 \rightarrow \mathbf{R}$ is defined as follows

$$\text{sim}_s(C_1, C_2) = w_0 \cdot \text{str}(C_1, C_2) + w_1 \cdot \text{map}(C_1, C_2)$$

where

$$\text{str}(C_1, C_2) = \sum_{(RC_1, RC_2) \subseteq RE(C_1, C_2)} S_s(RC_1, RC_2)^\alpha \cdot IC(RC_1, RC_2)\sqrt{\text{lensum}(RC_1, RC_2)}^\beta$$

with $\alpha, \beta \in [0, 1]$. $IC(C_1, C_2) = \sqrt{\log(p(C_1) \cdot \log(p(C_2))}$ and $\text{map}(C_1, C_2)$ is a similarity matrix computed by the use of the joint probability distributions determined by the related module described before.

The $IC$ function is generalized considering the estimated probability $p(C) = (\text{freq}(C)) + M \mu) / (|N_2| + M)$, where the Laplace estimator technique (for some $M \in \mathbf{N}, \mu \in [0, 1]$) is used to avoid values for which the function or its logarithm is undefined and $\text{freq}(C)$ returns the number of concepts linked via a path with $C$.

### 3.2 Linguistic and Contextual Validation

The uncertain mappings can be analyzed adopting a semantic viewpoint, based on external resources such as WordNet [15].

#### 3.2.1 Linguistic Validation

The linguistic validation module aims at comparing entities (classes and properties, relations) at a linguistic level. Entities are denoted by names that may differ, e.g., for the writing conventions or the terms used. Hence, the linguistic validation module requires named entities, say $E_1$ and $E_2$, whose similarity has to be assessed. The validation algorithm is made of two steps:

- **lexical analysis**: the name string is examined and divided into tokens based on a set of rules which specify how to detect explicit or implicit internal delimiters (e.g. a change in the capitalization: CompactDisc
yields the tokens \([\text{Compact, Disc}]\); these rules produce the token vectors \([t_{11}, \ldots, t_{1n}]\) and \([t_{21}, \ldots, t_{2m}]\); 

- **similarity assessment**: the \(n \times m\) matrix \(A_L = [a_{ij}]\) (linguistic affinity matrix) is obtained from the two vectors using WordNet, with \(a_{ij} = \text{SIM}_{\text{wn}}(t_{1j}, t_{2j})\) (see below). Then the linguistic similarity between the entities is computed by:

\[
\text{SIM}_L(E_1, E_2) = \frac{\sum_{j=1}^{m} \max_{i=1}^{n} a_{ij} + \sum_{i=1}^{n} \max_{j=1}^{m} a_{ij}}{n + m}
\]

The linguistic affinity of two entities exploits the semantics of the words exploiting their terminological relationships as provided by WordNet. These serve to create paths that link the terms to be compared. The first step finds the set of paths \(P_{1 \leftrightarrow 2}\) between the two terms (otherwise the similarity is null). Then, the strength of the paths is measured in order to select the best one. This is given in terms of the terminological relationships that are involved. Each kind of relationship may have its weight so that, say, synonymy is stronger than meronymy. The default weights considered in the module are:

<table>
<thead>
<tr>
<th>relationship (r)</th>
<th>weight (w_r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>synonymy</td>
<td>1.0</td>
</tr>
<tr>
<td>iponymy</td>
<td>0.8</td>
</tr>
<tr>
<td>iperonymy</td>
<td>0.8</td>
</tr>
<tr>
<td>meronymy</td>
<td>0.5</td>
</tr>
<tr>
<td>others</td>
<td>0.3</td>
</tr>
</tbody>
</table>

The strength of a path \(p\) is computed as the product of the weights of the relationships \(r_k\) that occur in it. The highest strength value determines the similarity of the two terms:

\[
\text{SIM}_{\text{wn}}(t_1, t_2) = \max_{p \in P_{1 \leftrightarrow 2}} \prod_{r_k \in p} w_{r_k}
\]

### 3.2.2 Contextual Validation

The contextual validation module combines the linguistic information discovered in the previous phase with relational aspects of the given concepts \((C_1\) and \(C_2)\) under comparison. The module uses the two ontologies \(O_1\) and \(O_2\) (reduced to a unique model which allows for language-independence) and the linguistic affinity matrix \(A_L\) previously computed combined with a relational affinity matrix \(A_R\) to produce a similarity value for the concepts under comparison. This new matrix encodes the kind of relationship holding between entities related to the two concepts.

The model the ontologies are reduced to is made up of a triple \(M = (\mathcal{C}, \mathcal{P}, \mathcal{R})\), where \(\mathcal{C}\) denotes the set of concepts, \(\mathcal{P}\) stands for the set of properties (unary relations) of the concepts, and \(\mathcal{R}\) contains the (binary) relations between concepts (allowed relations are same-as, kind-of, part-of, generic, with their natural semantics).

In the computation of the similarity, concepts and properties which are directly related to \(C_1\) and \(C_2\) are considered. Their names are used to compute linguistic affinity matrix, as previously illustrated. The computation of the relational affinity matrix \(A_R\) considers the kind of relation directly linking the concepts to the other entities in the respective ontology. The contextual relations are assigned with different default weights, as follows:

<table>
<thead>
<tr>
<th>relation</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>1.0</td>
</tr>
<tr>
<td>same-as</td>
<td>1.0</td>
</tr>
<tr>
<td>kind-of</td>
<td>0.8</td>
</tr>
<tr>
<td>part-of</td>
<td>0.7</td>
</tr>
<tr>
<td>generic</td>
<td>0.3</td>
</tr>
</tbody>
</table>

The relational similarity between two relations (properties) \(r_1\) and \(r_2\), with weights, resp., \(w_1\) and \(w_2\) is \(\text{SIM}_{\text{rel}}(r_1, r_2) = 1 - |w_1 - w_2|\). Hence the similarity of all couples of relations (properties) directly related to either concept is computed to fill the relational affinity matrix \(A_R\).

Finally, a measure of the contextual similarity is obtained. The contexts of the two concepts are elicited (entities that are directly related) as represented in the unique model: \(c(C_1) = \langle e_1^1, \ldots, e_1^n \rangle\), \(c(C_2) = \langle e_2^1, \ldots, e_2^n \rangle\), where each \(e = (s, r)\), i.e. the name string and the kind of entity involved. The contextual similarity is computed as follows:

\[
\text{SIM}_C(C_1, C_2) = \sum_{(t_1, t_2) \in c(C_1)} \sum_{(t_2, t_2) \in c(C_2)} A_L(t_1, t_2) \cdot A_R(t_1, t_2)
\]

### 3.3 Aggregation

Similarity values computed according to different perspectives (matchers) ought to be aggregated to provide a final value. Various methods have been proposed to compute such aggregate values (see [7, 1, 33]). All require the computation of weights (confidence levels) for the different matchers implemented. The main problem is then how to determine these values automatically (e.g. by recurring to machine learning techniques). One possibility is the usage of an ad hoc ensemble learning technique like stacking [36]. The problem with these methods is the amount of data required for training the individual and the meta-learner.

We resort to Yager’s OWA (Ordered Weights Aggregation) operators [22].

**Definition 1** (OWA Operator). Given a \(n\)-tuple of normalized weights \(\vec{w}\), an OWA operator is a function \(F : [0,1]^n \rightarrow [0,1]\) such that: \(F(a_1, \ldots, a_n) = \sum_{i=1}^{n} w_i b_i\) where the tuple \((b_1, \ldots, b_n)\) is obtained from \((a_1, \ldots, a_n)\) by sorting its elements in descending order.

Note that weights are associated to the positions rather than to the values.

The aggregation operator is based on a vector of weights computed through linguistic quantifiers [17]. A quantifier \(\theta\) can be represented as \(Q \subseteq I = [0,1]\), where for each \(r \in I\), \(Q(r)\) indicates the degree of satisfaction of the criterion specified by \(\theta\). So if \(\theta = \text{most}\) then \(Q(0.8) = 1\) means that 80% of the objects are compatible with this criterion. Suppose that \(n\) matchers produce \(n\) similarity values for the
AGGREGATION($\text{Cube}, \theta$)
input: $\text{Cube}$: similarity cube; $\theta$: quantifier;
output: $S_{\theta}$: $R^{n \times n}$; // final similarity matrix
begin
Compute weight vector $\vec{w}$ according to $\theta$;
for each candidate mapping $(C_1, C_2)$ do
begin
for $i \leftarrow 1$ to $m$ do
begin
$S_i \leftarrow \text{extract}(\text{Cube}, i)$;
$a_i \leftarrow S_i(C_1, C_2)$;
end
$\vec{a} \leftarrow \text{sort}(\vec{a})$; // sort in descending order
$S(C_1, C_2)_c \leftarrow \theta(S_1, S_2)_c = F_{\theta}(\vec{a})$;
end
return $S_{\theta}$;
end

Figure 5: Aggregation algorithm.

The aggregation process consists of the following steps (see Figures 4):

1. get the $m$ similarity matrices from the validation modules;
2. reduce them to sparse ones such that each concept $C_1$ in one ontology corresponds to a single $C_2$ in the other;
3. create a similarity cube by composition of such matrices;
4. given a selected quantifier $\theta$: compute the final similarity matrix based on the associated function (see algorithm illustrated in Fig. 5).

4. EXPERIMENTS
The experiments evaluated the improvement yielded by the adoption of the taxonomic and structural validators with respect to the performance of the base system. Moreover, we also wanted to find the related best linguistic quantifier.

4.1 Setup
Some of ontologies from the OAEI campaign\textsuperscript{7} were selected. Specifically the suite was made up of a reference ontology (101) and a number of variants obtained by omitting properties, replacing names with synonyms, changing the graph structures, etc. numbered 204, 205, 206, 221, 222, 223, 228, 233, 239, 240, 301, 304.

In order to compute the initial mappings, the four learners were set to their default parameters: Bayesian Name (n) and Bayesian Content (c) learners, k-Nearest Neighbor (k); Artificial Neural Network (a). Various combinations of the base-learners were possible. We will show the best ones, obtained by employing a choice of 3 or all four learners. As a similarity measure for computing the similarity matrix, the Jaccard similarity measure has been used.

Concept similarity was computed by the taxonomic validation module assuming the values computed by the system front-end (weights .3 and .4). The resulting mappings are selected when above of a threshold of .55. The Structural validation module was run with weights .3 and .4, resp., and maxlen = 12. The threshold for the selection of the results was set to .5. As regards the experiments with the linguistic and contextual validation modules, the similarity assessed using either approach was combined with even weights. Finally, the aggregation operator was tested using the presented linguistic quantifiers.

As metrics for evaluating the validity of the obtained mappings, precision, recall and F-measure as defined in [11] have been adopted.

4.2 Taxonomic Validator
In Tables 1, 2, and 3 the results of the base matching system (bs) are compared to those of the taxonomic validation (tv) in terms of precision, recall and F-measure, varying the choice of basic-learners.

The table 3 shows that tv improves with respect to bs in all but a couple of ontologies – namely, 221 and 233 – which have been obtained by eliminating the hierarchy. The improvement reaches the 15% for ontologies 223 and 240. Even more so, the improvement for 240 has been observed combining only 3 base-learners. This was probably due to the fact that the ontologies present a richer taxonomic structure than the original one. Conversely, for the same reason, slight decreases were observed for ontologies 222 and 239 because

\textsuperscript{7}http://oaei.ontologymatching.org
of their poorer taxonomic structures.

In terms of precision and recall the \( sv \) generally did not yield an improvement of the precision, as it may suggest erroneous mappings. Besides, the precision of \( bs \) with the ensemble of learners is already quite high, hence difficult to improve. The improvement is much more evident in terms of recall, as \( bs \) is not equally efficient.

### 4.3 Structural Validator

Tables 4, 5, 6 report the outcomes of the experiments, in terms of precision, recall and F-measure, comparing the results of \( (bs) \) to those of the structural validation \( (sv) \) varying the choice of basic-learners.

Table 4: Comparing the structural validator \( (sv) \) to the base system \( (bs) \) in terms of precision.

Table 5: Comparing the structural validator \( (sv) \) to the base system \( (bs) \) in terms of recall.

Table 6 shows that \( sv \) improves the performance of the system, except for the ontology 233 for the same reasons outlined before. This was evident especially for ontologies 223, 239, and 240 (with a peak of +19% for 239) with the combination \( k-a-c \). In particular, ontology 223 presents an extensive hierarchy which helped finding related concepts. A little decay was observed for ontology 221, one where the taxonomic was eliminated w.r.t. the original one, with opposite effects compared to the mentioned 223.

Again, disaggregating these outcomes in terms of precision and recall, one observes that precision of \( bs \) was already quite high and so difficult to be further improved: a single erroneously validated candidate mapping may even worsen the precision. Recall is improved by the validator w.r.t. the
base system up to a 31% observed on ontology 239 (with the k-a-c combination) although the restriction of the hierarchical structure of this ontology w.r.t. the original one and the elimination of the properties might lead to predict this as a difficult case for sv.

### 4.4 Linguistic + Contextual Validators

Another experiment evaluated the performance of the other validators by combining the decisions made using both the linguistic validation and contextual validation modules (lcv) and comparing them to those made by base matching system (bs), with the same combinations of basic learners, in terms of precision, recall and F-measure. Tables 7, 8 and 9 report the outcomes of such experiments.

Starting from the evaluation of the F-measure results (see Table 9), it is evident an improvement for almost all of the ontologies (11 out of 12). This is mostly due to an improved recall with large improvements in some cases and only overall performance of ao. The real gain is then in terms of recall with large improvements in some cases and only two cases were a minimal decay was observed (222 and 223) w.r.t. the performance of ao.

### 4.5 Aggregation Operator

Finally, we made experiments for testing bs together with the various additional components of the system, and especially the aggregation operator. Preliminarily, we tested the linguistic quantifiers. This produced a choice of the best quantifier to be utilized for the aggregation operator in the comparison with the other components (see Tab. 10). This has led to selecting the Max quantifier, although also Alh, Few and Nh produced good results.

Then experiments were performed comparing bs, tv, sv, lcv and aggregation (ao). Table 11, reporting the average F-measure outcomes, shows that using ao the system was often able to produce better results. The most problematic ontologies were 222, 223, 239 and 301 for which ao lost some correct mappings. However, in general, ao produced the maximum average improvement w.r.t. the performance of bs. As with the previous experiments with the validators, the values for the precision index were difficult to improve.

### 5. RELATED WORK

The two main approaches that are used for coping with the ontology matching problem are: 1) the schema-level approach (only the ontology schema is considered); 2) the element-and-structure-level approach (ontology instances are also taken into account) [12]. Most of the existing works focus on the schema-level approach and only few of them also consider ontology instances. Indeed, many works have addressed the problem in the context of ontology design and integration [27, 28, 25]. The initial trend has been to set up elementary matchers copying with a particular aspect relevant. By comparing the average values of the F-measure over all of the ontologies the next increase when using the validators is evident, especially the cases of test ontologies obtained by hierarchy variations and synonym replacements.
In the last years, the idea of combining individual matchers has been pursued by many existing ontology matching approaches (e.g., see [6, 1, 21]). The points of difference regard the employed features and the way the results of the individual matchers are combined. In contrast to other proposed approaches that combine a number of specific predefined classifiers, our approach is a more general one, as we do not make any assumptions about the individual matchers to be combined apart from the fact that they provide their results in a standardized format.

In the MoTo system there are two levels of combination: one regards the stacking of base-learners which is used to discern certain mappings from the others; the other regards the recovering of uncertain mappings through additional structural and linguistic validation mechanisms whose output can be finally aggregated by means of specific operators.

In GLUE [8] a meta-learning approach is applied for generating matching hypotheses on the basis of multiple local classifiers trained on different aspects of the models to be matched. This approach, requires that the input for meta-learning is generated by specific probabilistic learning methods (naive Bayes classifiers), that are integrated using a linear combination at the meta-level. In our case, we do not make any assumptions about the matchers used, as it is possible, in principle, to apply any machine learning technique at the meta-level. This makes our approach more general.

Besides, the evaluation in [8] is performed on a rather limited set of ontologies without the existence of commonly agreed reference alignments. As pointed out in [9], the results of that meta-learning approach is dominated by the so-called content learner, which always performs almost as good as the integrated learning approach making the meta-learning step less important. As reported in the next section, validation and aggregation perform often significantly better than any of the local matchers.

In [20] kernel machines are used to learn a classifier for mapping correctness based on a set of simple similarity measures. The classifier is evaluated on the benchmark datasets of the OAEI and outperforms existing matching systems. The setting of the experiments has been criticized in [9], as being rather unrealistic since all existing reference mappings are thrown together in one large training set and cross-validation is used for computing the accuracy of the classifier. Further, the approach uses only similarity values as input for learning.

In the system described in [32] various Bayesian classifiers are employed. However, the feature set is based on string distance measures typically adopted by matching systems for determining a syntactic similarity. The experimental results show that there is a strong correlation between different measures and the machine learning approach cannot significantly improve the results of the best individual measure. As shown in [9], using additional and diverse features can significantly improve the classification result.

In [24] decision trees and rule learners are used to learn rules for integrating the results of different matching systems. Confidence values respectively measured similarities as the basis for learning. The approach has been evaluated on a subset of the OAEI benchmark dataset but learning from confidence values only does not seem to be a good choice [9].

As regards the aggregation operator, we have followed the line of COMA [6] and COMA++ [1] implementing more linguistic quantifiers. The CMC system [33] was meant to overcome the limitations of LSD and COMA by predicting the credibility of the individual matchers and then combining (couplewise) the resulting similarity matrices by means of the same method adopted by COMA.

6. CONCLUSIONS

This work focused on ontology matching validation based on structural aspects on the ontologies. It also concerned the aggregation of the similarities computed by different matchers, that are able to reconcile the various aspects targeted by each matcher through many linguistic quantifiers. The experimentation demonstrates that a combination of different techniques yields some added value in the quality of mappings found. Specifically, the taxonomic validator proved its effectiveness despite of the simplicity of the underlying idea. Another point in favor is that it can be applied to any kind of ontology being focused on the taxonomic aspect only. The structural validator goes a step even further as it exploits also other relationships between the concepts. The aggregation operator can select the best mappings from each system component and allows different types of aggregation by changing the most appropriate quantifier.

We are currently planning an enhancement of the validators so that other criteria may be implemented. The structural validator may also be enhanced by taking into account annotations/comments in natural language. A deeper investigation of the application of ensemble machine methods is also necessary. This may affect also the choice of weights for the aggregation operator.

| Table 11: Comparing the base system (bs) and the validators to the aggregation operator (ao). |
|---|---|---|---|---|---|
| bs | lcv | tv | sv | ao |
| 204 | .91 | .91 | .92 | .92 |
| 205 | .62 | .56 | .57 | .61 |
| 206 | .66 | .64 | .68 | .65 |
| 221 | .66 | .61 | .63 | .63 |
| 222 | .93 | .98 | .93 | .99 |
| 223 | .87 | .95 | .92 | .95 |
| 228 | .94 | .96 | .98 | .97 |
| 233 | .40 | .52 | .40 | .40 |
| 239 | .84 | .91 | .88 | .95 |
| 240 | .83 | .86 | .90 | .88 |
| 301 | .61 | .61 | .63 | .66 |
| 304 | .76 | .83 | .80 | .86 | .88 |
7. ACKNOWLEDGMENTS
The authors wish to thank C. Leone, M. Tempesta and C. Terlizzi who designed and implemented the system MoTo.

8. REFERENCES


About the authors:

Nicola Fanizzi (PhD in Computer Science), is assistant professor at the Computer Science Dept. of the University of Bari since 2001. The research area of interest, within the broad field of Artificial Intelligence, is Machine Learning and Knowledge Discovery, particularly the methodologies for complex (multi-relational) representations based on fragments of First-Order Logics. Currently he is working on Semantic-Web mining methodologies, focusing on the problems of classification, clustering, ranking, and uncertainty reasoning. This work was/is being carried out within national as well as international research projects and networks of excellence. He authored 100+ scientific papers appeared on scientific journals, proceedings of national and international conferences and workshops. He served in the OC/PC of several workshops and conferences related to Artificial Intelligence, Machine Learning, Data/Web Mining and to the context of the Semantic Web research. See also www.di.uniba.it/~fanizzi

Claudia d'Amato (PhD in Computer Science), is a research fellow at the University of Bari - Computer Science Department. Her main topics of interest are: similarity measures for concepts and individuals in Description Logics, supervised and unsupervised methods for ontology mining, approximate and uncertain reasoning for the Semantic Web. She is author of more that 50 papers among international collections and journals. She has been invited expert for the W3C Uncertainty Reasoning for the World Wide Web Incubator Group. She is organizer of the international Uncertainty Reasoning for the Semantic Web workshop and of the Workshop on Inductive Reasoning and Machine Learning for the Semantic Web. She was Vice-Chair for ISWC 2009, she is editorial board member of the Semantic Web Journal and she is also serving as program committee member in international conferences such as ISWC’10, AAAI’10, ECAI’10, ECML/PKDD’10, SAC’10, EKAW’10, RR’10, RuleML’10, WI’10, STAIRS’10, ICAI’10, KARE’10. She also received the nomination from the Italian Artificial Intelligence community for her PhD thesis “Similarity-based learning methods for the Semantic Web” as one of the best Italian PhD contribution in the area.

Floriana Esposito is author of more than 250 papers which have been published in international journals and Proceedings. On August 2006 she received the ECCAI Fellowship in recognition for Pioneering Work in the field of Artificial Intelligence.

Since 1984 associate professor of Computer Science at University of Bari and since 1994 Full Professor of Computer Science, Chair of “Knowledge Engineering”.

Her scientific interests are in the area of Artificial Intelligence and concern the logical and algebraic foundations of Machine Learning: more in particular, the integration of numerical and symbolic methods in learning and classification, the logical foundations of conceptual learning techniques, the automated revision of logical theories, multi-strategy learning for discovery in data bases, inductive logic programming. The major application fields are Document processing and Digital Libraries and, recently, Bioinformatics.
A Robust Link-Translating Proxy Server Mirroring the Whole Web

Ziqing Mao
Purdue University
305 N. University Street
West Lafayette, IN
zmao@purdue.edu

Cormac Herley
Microsoft Research
One Microsoft Way
Redmond, WA
cormac@microsoft.com

ABSTRACT

Link-translating proxies are widely used for anonymous browsing, policy circumvention and WebVPN functions. These are implemented by encoding the destination URL in the path of the URL submitted to the proxy and rewriting all links before returning the document to the client. Commonly these are based on the CGIProxy implementation or a variant. While popular, broken links and very slow load times are common. This is so, since the use of scripting languages makes finding and translating links (the essential task of such a proxy) very difficult. Some web-sites become entirely non-functional when loaded.

This paper proposes a novel architecture for a link-translating proxy. By using a sub-domain mapping technique we entirely eliminate the need to translate (or even find) relative links in content. This makes the link rewriting algorithm very simple and robust. We have implemented, deployed, and tested the proxy extensively. We conducted a comparison between the our proxy and CGIProxy. Both the theoretical analysis and experimental evaluation show that our proxy is significantly better in terms of robustness, performance, and security. We call our system Address Bar Only Proxy or ABOProxy. We suggest several novel applications.

1. INTRODUCTION

Web proxy servers perform a critical role for many organizations. They provide caching, acceleration, content filtering and isolation functions. Most of this is invisible to users who are unaware whether any proxies intervene between them and the end server. Commonly there is a web proxy that the browser uses for outbound web traffic; this is set and controlled by the ISP or home network, and is used for caching, policy enforcement, auditing and so on. This is often known as a forward proxy. In contrast, a reverse proxy generally sits close to a server and dispatches in-bound web traffic to a set of backend servers, presenting a single interface to the clients [20]. It is generally used to pass requests from the Internet, through a firewall to an isolated, private network, preventing Internet clients having direct, un-monitored access to the sensitive servers. Besides forward and reverse proxies, sometimes an intermediate proxy sits between them. This is typically provided by a third-party in the Internet; (sometimes this is unintentional where a mis-configured server acts as an open relay [1, 2]). In each of these three cases the proxy intercepts the traffic at the HTTP layer or lower. Forward proxy settings are often auto-detected by the browser (e.g. using the Web Proxy Auto Detect (WPAD) service in Windows [24]). When using a reverse proxy the backend servers are configured to accept and send all traffic through the reverse proxy. An intermediate proxy is often reached by changing the proxy settings on the browser manually. Figure 1 summarizes the different kinds of proxy servers.

The proxy architecture discussed in this paper differs from these three kinds of proxy. We call it a web-service based or link-translating proxy, because the proxy server is running and is reachable as a standard web-service. In terms of network location, the link-translating proxy is similar to an intermediate proxy. However, its main advantage is that no alteration of browser or other settings is required. To use the proxy, the user submits an initial desired URL in a form or in the address bar without changing proxy settings. The proxy returns the webpage of the destination URL and all communication for the subsequent clicks and actions performed in the webpage will also go through the proxy. Such proxies are widely used [3]. The most popular link-translating proxy is an open-source implementation called CGIProxy [4], which is often used in anonymizing proxies. Another use is in webVPN products, such as Cisco’s. We discuss the technique used by the existing solutions and its shortcomings in Section 3.1.

In this paper, we propose a novel web-service based link-translating proxy. We call it Address-Bar-Only Proxy (ABOProxy), because the user simply uses the address-bar to direct traffic through the proxy. We
borrow an elegant idea first used in the design of the Coral Content Delivery Network [21] and adapt it to the problem addressed by CGIPProxy [4]. Our design outperforms the CGIPProxy in design simplicity, robustness, security, load time and scalability. To use ABOPProxy, the user simply appends .aboproxy.com to the end of the URL hostname. For example, to visit www.google.com through the proxy, the user navigates to www.google.com.aboproxy.com. All the communication for the subsequent clicks and actions performed in the webpage will also go through the proxy (whether fetched from the same domain or other domains). We describe our method in Section 3. The implementation and evaluation are presented in Section 4 and Section 5, respectively.

2. RELATED WORK

We now review several related architectures. An excellent text on web proxies is [20].

Coral. An important building block of our system is borrowed from the design of the Coral [21]. Coral is a peer-to-peer content distribution network that enables web developers to offer high performance and meet huge demand without large infrastructure investment. To reach content via the network a user appends .nyud.net to an address. The request is then directed to a nearby Coral cache. Relative links within the response will automatically point back to Coral (just as with our subdomain mapping described in Section 3.2.1). However Coral makes no effort to find or translate absolute links. Thus absolute links, and content that Coral decides not to cache will be fetched from the origin server. While we borrow a very important design element from Coral our goal is very different. While Coral seeks to improve the performance in loading content for relative links, we seek to provide a robust proxy for all links, and entire sessions, including SSL sessions. Thus the major differences between our proxy and Coral are as follows. We search for and translate absolute links, whether statically or dynamically encoded. Since we point at a single proxy rather than many caches our DNS configuration is simpler. We preserve session information as represented by cookies. Since, in Coral, relative links are fetched from .nyud.net, and absolute ones from the origin server, a cookie set in one will not be available to the other even though they belong to the same domain. Thus an artificial security boundary is constructed between content from the same origin site, which can break the experience. We handle both SSL and regular content. User applications that our proxy handles that Coral cannot include secure browsing, anonymous browsing, logging in to websites and any browsing that relies on cookies and dynamic content.

CGIPProxy. The most popular general link-translating proxy implementation is CGIPProxy [4]. This a generic link-translating proxy with implementations available for most major platforms. It’s primary uses seem to be to provide anonymous web-browsing and to circumvent network policies (e.g. surf to forbidden sites). Many of the popular circumventors are based on instances of this code (see e.g. [5]). There is an enormous number of instances of CGIPProxy running at any given time. For example [3] lists 5200 of what it claims are more than 26000 instances running. The major differences between our proxy and CGIPProxy are as follows. By using sub-domain mapping relative links resolve automatically. We experience far fewer broken links, load times are far faster, and the proxy is more scalable. We eliminate several security concerns such as cookie stealing and cross-site scripting (see Section 5.3).

Open Relays. An open relay is a proxy that will accept traffic from any client and relay it to any server. Often this occurs when an administrator mis-configures a machine. For example many firewall products, such as ISA [6] or UserGate [7] can be configured to proxy from anywhere to anywhere (though this is seldom what a system administrator would intend). There appears to be enormous demand for open relays. There are several sites [1, 2] that are dedicated to providing up to date lists of IP addresses and port numbers where open relays can be found. Thus users can manually configure their browser to use one of these open relays as proxy and have all their traffic appear to come from the relay rather than their own IP address. For example [2] lists what it claims are about 2000 new open relays per day.

Web VPN. Virtual Private Networks (VPNs) allow for secure communication to internal resources from external locations. Traditional VPNs require software to be installed on the client machine, while WebVPN is a VPN connection that is established using only the web browser.

Many universities uses WebVPN to enable the students to access restricted websites outside the campus. The WebVPN is achieved using a link-translating proxy which runs a web-service that fetches the internal web-pages for the clients. However, dynamic web content...
does not work well when being accessed through a WebVPN. For example, on the CMU WebVPN website, users are warned that 11 databases cannot be accessed with WebVPN. In addition, there are known cross-site scripting vulnerabilities [8] with the popular WebVPN products from Cisco. The details of the security problem is discussed in Section 5.3. Using the ABOProxy for WebVPN can robustly serve any web content, and eliminates the security problem.

Debugging Proxies. There are a number of lightweight proxy applications that are suitable for use by a single or small number of users. Generally these are not intended to scale to supporting a large number of users. For example Charles [9] and Fiddler [10] are debugging proxies mainly intended to enable developers to examine the request/responses stream that flows between the browser and web-site. They provide powerful tools to examine, modify and replay headers, content and cookies in great detail. Both provide interfaces to allow modification of the request/response stream, and handle SSL as well as regular HTTP traffic. It is thus possible using either of these proxies to customize the browsing experience. Burp [11] and Paros [12] provide similar functionality.

Personal proxy. Personal proxies have been widely used for debugging, form-filling, content-filtering, archiving browsing history, ad-removal and so on [13]. There are a great number of open-source and product personal proxies. All of them require the user to manually change the browser’s proxy settings in order to use the proxy from other machines. Using the ABOProxy eliminates this requirement. An advantage of ABOProxy is that a user might get all of the advantages of a personal proxy from any location. That is, even when alteration of the browser proxy settings is not possible or convenient the functionality would still be reachable. Various filtering functions suitable for personal proxies are suggested by the Internet Content Adaptation Forum [14]. Usually the browser is configured to use either a debugging or personal proxy. Thus the entire question of link translation (i.e., the main subject of our work) does not arise.

Other Uses of Proxies Gabber et al. [19] describe a proxy system, known as the Lucent Personalized Web Assistant, which acts as a credential intermediary and allows anonymous access to various web-services. The user's browser is configured to use the proxy, which might reside on their own machine or in the cloud.

Provos and Holz [22] describe a creative use of a proxy which monitors the loading of blacklisted sites. The idea is that vulnerable web-servers can be made to serve content that loads from malicious sites. SpyBye [22] acts as a proxy and monitors all of the requests originating from the browser. By maintaining a blacklist and a set of URL classification rules it can determine when fetches are sent to malicious or unapproved servers.

The Tor network [23, 15] also offers a means for Internet users to conceal their browsing patterns even from traffic analysis.

3. METHOD

3.1 Background and Existing Approaches

The most popular and reliable link-translating proxy implementation is CGIProxy [4], which is a server-side script in Perl and has been developed and actively updated since 1998. WebVPN’s use very similar link translation techniques. The general idea is to encode the destination URL in the path. The idea can be illustrated with an example. Suppose the CGIProxy’s script is running at the address http://cgiproxy.com/nph-proxy.cgi. If the user wants to visit http://www.google.com through the CGIProxy, then the URL shown in the browser’s address-bar is http://cgiproxy.com/nph-proxy.cgi/http/www.google.com. In this way, when the perl script receives the HTTP request, it is able to decode the destination URL from the path of the submitted URL. More specifically, the script does the following upon receiving each request.

1. Decode the destination URL from the path of the URL.
2. Retrieve the content from the destination website.
3. Modify the response by rewriting all links, pointing them back to the proxy script.
4. Send the modified response to the user.

The key operation of the proxy script is the link rewriting in Step 3. The purpose of which is to point all links contained in the server’s response back to the proxy, so that when those links are triggered in the user’s browser, the requests will be also routed through the proxy. For example, in Google’s homepage, there is a logo image file embed by the following tag (img src="/intl/en_ALL/images/logo.gif"). The proxy script has to modify the link of the logo image to be /nph-proxy.cgi/http/www.google.com/intl/en_ALL/images/logo.gif. The design is conceptually simple and clear. However, it is actually very difficult to perform link rewriting robustly.

3.1.1 Understanding Links

In HTML a link can be either absolute or relative. An absolute link defines the location absolutely including the protocol, the host-name, and the path of the re-
Table 1: The categorization of links and the hardness of link rewriting

<table>
<thead>
<tr>
<th></th>
<th>statically encoded</th>
<th>dynamically generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>absolute links</td>
<td>easy</td>
<td>hard</td>
</tr>
<tr>
<td>relative links</td>
<td>easy</td>
<td>hard</td>
</tr>
</tbody>
</table>

Note: In the standard definition, an absolute URL takes the form `protocol://hostname/path?query_parameters`. To make the description simpler, in this paper we treat the query parameters as a part of the path.

generated links in an ad-hoc way. Instead of implementing a complete JavaScript engine, the CGIProxy uses heuristics to identify the constants and variables that possibly store links, and modify those links by invoking a function. For example, the function call `window.open` opens a webpage in a new or an existing window. The first argument of the function call specifies the URL, such as `u="next.html"`; `window.open(u)`. The CGIProxy identifies the variable `u` stores a URL and modify the JavaScript to `u="next.html"`; `window.open(proxify(u))`, where `proxify` is a function provided by the CGIProxy to rewrite an individual link. Consider, the example where the website has the JavaScript: `u="next.html"; w=document.getElementById("sub_frame"); w.contentWindow.open(u);`.

In order to capture the link stored in `u`, the proxy has to identify the DOM element with ID `sub_frame` has an attribute named `contentWindow` that implements the interface `window`. This can not be done without a complete HTML and JavaScript engine. This example merely shows the difficulty of finding dynamically generated links; there certainly exist many other cases making the problem very complicated (in fact in general undecidable).

3.2 A Novel Approach

3.2.1 The Key Idea: Subdomain-Mapping

The key idea of our approach is to map each destination domain to a sub-domain of the proxy server by appending the proxy name to the destination hostname. For example, the proxy server is running at `aboproxy.com`, so the domain `www.google.com` is mapped to `www.google.com.aboproxy.com`. By navigating to `www.google.com.aboproxy.com`, the user visits the Google homepage through the proxy. Similarly, the logo image would be retrieved from `http://www.google.com.aboproxy.com/intl/en_ALL/images/logo.gif`. We call this technique `subdomain-mapping`, which simply appends `.aboproxy.com` to the end of the original URL hostname. We will see that this allows more robust and complete link translation.

3.2.2 DNS Configuration

The Domain Name System (DNS) provides the service to map between domain names and IP addresses. Each domain or sub-domain has one or more authoritative DNS server, which maintains a list of DNS records. Each DNS record maps a domain name to an IP address. In the DNS server for the proxy domain, we use a wildcard DNS record to map the domain `*.aboproxy.com` to the IP address of the server machine. Such a record will match all requests for domain names ending in `aboproxy.com`. As a result, all
requests for a sub-domain within aboproxy.com will be delivered to the server machine of the proxy.

3.2.3 Server-side Script

Our proxy runs a script. On receiving each request, the script performs the following operations.

1. Decode the destination URL by removing .aboproxy.com from the URL in the request.
2. Retrieve the contents from the destination website.
3. Modify all absolute links in the response, by appending .aboproxy.com to the URL hostname, pointing them back to the proxy.
4. Send the modified response to the user.

Again, the key step for the proxy script is to do the link rewriting in server’s responses. Observe that only absolute links need be translated. This is a main point of contrast with other web-service proxies.

3.2.4 Robust Absolute Link Rewriting

First, the path component of the URL remains unchanged after the subdomain-mapping. Therefore, all relative links, whether statically or dynamically generated, do not need modification. The subdomain-mapping also makes rewriting absolute links easy. We merely append .aboproxy.com to the end of the URL hostname. As discussed before, the real difficulty of link rewriting comes from finding links that are dynamically generated by client-side scripts. We have eliminated the need to modify relative links. For absolute links, instead of trying to find all variables and constants that store links, we just need to find all positions that are the end of hostnames. We use pattern matching to do that.

Recall that hostnames end with a fixed set of tails, such as .com, .edu, and so on. They are called top-level domains. There are two groups of top-level domains, generic top-level domains and country-code top-level domains. Generic top-level domains are maintained by Internet Assigned Numbers Authority (IANA), and there are currently 33 generic top-level domains, including .com, .edu, .net, .org, and so on. Country-code top-level domains are used or reversed for a country or a dependent territory. There are currently 253 country-code top-level domains as described in ISO 3166. A complete list of top-level domains is available officially at http://www.iana.org/domains/root/db/. With the set of all top-level domains, by searching those patterns in the server’s response, we can locate a set of candidate positions that may be the end of hostnames. However, certainly not all the appearances of the top-level domain patterns are really the end of hostnames. We need to do a further selection by examining the characters follows the appearances of top-level domain patterns. There are three cases that follow a hostname.

1. The hostname is followed by the port number, connected by the character ‘:’. For example, https://www.vanguard.com:443.
2. The hostname is followed by the path, connected by the character ‘/’. For example, http://www.google.com/services/.
3. The hostname is the end of a URL, or a string constant, which are identified by the characters ‘‘’ and “"”. The string constant may be further used in calculating URLs.

Therefore, only when the character immediately follows the appearance of top-level domain pattern is one of the four characters: : / ‘ ”, is it identified as the end of a hostname. Figure 2 gives some examples on when to and when not to identify .com as the end of a hostname. Last, we append .aboproxy.com at all the positions that are identified as the end of hostnames in the server’s response.

Limitations

There are a few limitations of the link rewriting algorithm described above. First, it is possible to fail to find a hostname and thus not translate a link. In fact, as we have said, the problem of finding all dynamically generated links is generally unsolvable. Second, it is possible to translate something that appears to be a hostname, and yet is not a link.

Of course it is certainly possible for a web-site to break the algorithm (i.e. conceal links from the proxy script) intentionally. There exist two ways, (1) hide the top-level-domain pattern; (2) add a character other than those : / ‘ “ to the end of a hostname. The examples include:

```
u1="http://www.example." + "c" + "+" + "m";
u2="http://www.example.cpp"; u2=u2.replace(/pp/, "+om");
u3="http://www.example.com"; u3=u3.substr(0, u.length-1);
```

However, these techniques are adversarial rather than normal: they are essentially unknown in benign real-world web applications (see Section 5.1). However, this does imply that ABOProxy will not necessarily function well in adversarial settings (a good example of a proxy in an adversarial setting is SpyBye [22]).

Since they do not end with generic or country code top-level domains, our implementation makes no attempt to translate numeric IP address links. This has minor impact since such links are now rare. However a user who loads a page from the Google search cache (which loads from numeric addresses) will go directly to the cache rather than the proxy.

Second, the algorithm may append .aboproxy.com to a hostname that will be displayed in the webpage. For example, the algorithm will append .aboproxy.com in the following JavaScript, `document.write("Thank you`
Examples

<table>
<thead>
<tr>
<th>src='<a href="http://www.example.com">http://www.example.com</a>'</th>
<th>The end of a hostname?</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>window.open('<a href="http://www.example.com/index.html">http://www.example.com/index.html</a>')</td>
<td>Yes</td>
<td>A statically encoded link</td>
</tr>
<tr>
<td>sTargetURL = &quot;<a href="http://collect.myspace.com/index.cfm?MyToken=">http://collect.myspace.com/index.cfm?MyToken=</a>&quot; + myToken;</td>
<td>Yes</td>
<td>A dynamically generated link</td>
</tr>
<tr>
<td>... if (b.xk==Hna) {b.Dt(Ina);a=b[a];return a+Jna}</td>
<td>Yes</td>
<td>A dynamically generated link</td>
</tr>
<tr>
<td>Sys.component=function() {...}</td>
<td>No</td>
<td>Use JavaScript to compute links (from GMail)</td>
</tr>
<tr>
<td>u='<a href="http://www.sina.com.cn">http://www.sina.com.cn</a>'</td>
<td>No</td>
<td>A method starting with &quot;-com&quot;</td>
</tr>
</tbody>
</table>

Table 2: Examples on when to and when not to append "aboproxy.com".

for visiting example.com"). We don’t try to distinguish between a hostname in a link and a hostname displayed in a webpage. The only effect is that the user will see a superfluous .aboproxy.com on the webpage and it doesn’t affect the functionality.

3.2.5 HTTP Cookies

HTTP Cookies are pieces of text data sent by a web server to a browser to maintain state. Each cookie is a name/value pair with four optional fields: expires, domain, path, and secure. The domain and path fields indicate for which HTTP requests the cookie should be sent back. For example, if the user requests the URL http://www.bank.com/credit/index.html, then a cookie with domain=.bank.com and path=/credit would be attached to the request, but a cookie with domain=script.bank.com would not. A server sends cookies together with their attribute values to a browser in the Set-Cookie header of the HTTP responses, and the browser sends the cookies back using the Cookie header of the HTTP requests. With the introduction of the proxy, from the browser’s point of view, the cookies are set by the proxy server returned to it. Therefore, the proxy needs to modify the attribute values. The subdomain-mapping technique makes the modification very simple. We append .aboproxy.com to the end of the original value of the attribute domain. In this way, as shown in Figure 2, the browser will send back the appropriate set of cookies for each request. We don’t need to modify the value of the attribute path because the path components in the URLs remain unchanged.

4. IMPLEMENTATION

We have implemented and deployed a prototype of ABOProy. The main requirement for the implementation is a machine that connects to the Internet with a web server installed. In addition we require registration and control of the DNS settings for a valid second-level domain name. If SSL content is to be fetched we also require a wildcard certificate.

4.1 Domain Registration and DNS Setup

We acquired the second-level domain aboproxy.com and custom DNS hosting service from DynDNS.com. In the DNS settings, we added a wildcard host record that maps *.aboproxy.com to the IP address of our server machine (this means that all requests ending in .aboproxy.com will be referred to that server). The total cost for the domain registration and DNS service is $42.5. Actually this can even be done without cost. Among others, DynDNS.com provides free dynamic DNS service. Instead of a paid-for second-level domain, a free third-level hostname is obtained (within 88 available second-level domains), e.g., aboproxy.dontexist.com. Next the wildcard option is enabled. Third, the IP address of the server machine is entered, so that all hostnames matching *.aboproxy.dontexist.com are mapped to the server’s IP. Figure 3 shows these changes being made for a free third-level domain at DynDNS. The procedure is similar for a paid-for second level domain. Other registrars provide similar functionality.
4.2 Web Server Setup

Our implementation is based on ASP.NET 2.0 and Microsoft IIS 6.0 running on Windows 2003 SP2. We implemented the proxy script as an HttpHandler in C#. HttpHandler is a .NET component that acts as a target for incoming HTTP requests. In other words, every incoming HTTP request will be delivered to the proxy script. The proxy script can be implemented using other server-side scripting languages, such as Perl and Python. Besides IIS, the server can use other web server applications that supports server-side scripting. For example, in the Apache web server, one can use the directive SetHandler to make all requests to be delivered to the scripting handler cgi-script, which will invoke the corresponding script interpreter.

4.3 SSL and Certificates

If the user visits a HTTPS website through the proxy, e.g., https://www.paypal.com.aboproxy.com, the proxy will maintain one SSL connection with the client browser and another SSL connection with the end server. In this way, the proxy server gets the response from the server from one secure connection, does the link rewriting and sends it back to the client over another secure connection. Obviously, all of the traffic is available un-encrypted to ABOProxy. To be able to proxy to any HTTPS website, we obtained a wildcard certificate for *.aboproxy.com. However, the handling of wildcard certificates (as governed by RFC 2818) is browser dependent. All versions of Firefox and Opera that we tested (i.e. Firefox 1.0 and variants and 2.0 and Opera 9.25) allow the wildcard to match multiple fields in the hostname. So, the certificate for *.aboproxy.com matches mail.google.com.aboproxy.com, www.paypal.com.aboproxy.com, etc. However, IE, Safari only allow using the wildcard to match the leftmost field in the hostname. In this way, the wildcard certificate for *.aboproxy.com would match a.aboproxy.com, or b.aboproxy.com, etc., but would not match a.b.aboproxy.com. In addition, they do not accept certificates with multiple wildcards, such as *.aboproxy.com. In conclusion, with a wildcard certificate, users can visit any HTTPS websites through the proxy without having any certificate warnings in Firefox and Opera. But in IE, Safari and Chrome, there will be certificate warnings in our current implementation.

We point out that the occasional certificate warnings are specific to our prototype implementation and there are several ways to rectify this. If the proxy is restricted to reaching a fixed set of websites, we can use a certificate for all those sub-domains by specifying multiple dNSName fields in the certificate. For example, if the proxy is restricted to reaching https://mail.google.com and https://www.paypal.com, then we would need a certificate with two dNSName fields of https://mail.google.com.aboproxy.com and https://www.paypal.com.aboproxy.com. It is possible to get certificates with a hundred or more dNSName entries, which makes it possible to cover a large number of sites. In this way, we can avoid certificate warnings in IE, Safari and Chrome. We have followed this approach for the one-time password system application in Section 6. Alternatively, we could get a separate certificate for each SSL domain to be reached. Neither IIS nor Apache support serving different certificates depending on the content.

5. EVALUATION AND STATUS

5.1 Efficacy

We tested the ABOProxy implementation by using it for everyday browsing. When typing in a URL in the address bar, we appended .aboproxy.com to the end of the URL hostname. In this way, all following traffic goes through the ABOProxy. In four months testing with two users (i.e. the authors) the ABOProxy provided the same experience as that without using the proxy. We tested using a variety of browsers and from locations in the US, Canada, Brazil, Taiwan, Ireland and Australia. Specifically our everyday browsing included the activities shown in Table 3. Use requires merely appending .aboproxy.com to any URL hostname, so that https://www.google.com becomes https://www.google.com.aboproxy.com etc. Note that this instance is for evaluation purposes only; we may restrict traffic or shut down the proxy without notice.

We only observed one set of cases where ABOProxy implementation failed. That is due to the extra restrictions that the IIS and .NET framework place on URL’s besides those specified in RFC 1738. The IIS and .NET framework have the following extra restrictions.

1. The total length of the path in the URL should not exceed 160.

2. The path in the URL should not contain characters that are not allowed to be used in the NTFS file names, such as :, *, |, and so on.

The first restriction is placed by the IIS, and the second is from the .NET framework. Both are un-configurable. We circumvented the second restriction.
by adding an extension before the .NET in the stack. The extension maps each of those special characters to a specific sequence of normal characters. The .NET extension can use the mapping rules to change those characters back when processing the requests.

We are unable to work around the first restriction in any IIS based implementation. The websites hosted using other web servers, such as Apache, may use a very long URL whose path is longer than 160. When visiting such a URL through the ABOProxy, the IIS server will respond a 404 (bad request) error. In our testing, we observed very few websites use URLs that have a path longer than 160. Only two exceptions were observed: doubleclick.com and yahoo.com. URLs of length greater than 160 even from these sites are extremely rare.

**CGIPProxy** For comparison, we also tested the efficacy of the CGIPProxy implementation. We set up the CGIPProxy version 2.1beta18 (released on Aug 10, 2008)\(^5\) with a Apache web server 2.2, and visited some popular websites through the proxy using Firefox 2.0. We observed many basic operations failed when performed through the proxy. The CGIPProxy broke the functionalities of those websites. A summary of the failure cases of CGIPProxy on some popular sites is given in Table 4.

### 5.2 Performance Evaluation

The following experiments were performed on a Server machine with Intel XEON Processor 2.40GHz, 1GB RAM and 90GB hard disk. The server machine is running Window Server 2003 SP2, IIS 6.0 and .NET Framework 2.0. For comparison, we conducted the same experiments against the CGIPProxy implementation on the same server machine and report the results.

#### 5.2.1 Page load time and page size

We visited a set of popular web site through the proxies using Firefox. We measured the time used to load the web pages and the total amount of HTTP data received by the browser to load the pages. Before each visit, we removed all cache files in both the server machine and the client machine. The page load time is measured using the Load Time Analyze extension, and the page size is measured using the FireBug extension.

The evaluation results is shown in Figure 4. The results for “conventional web proxy” are obtained by configuring the browser to use a standard web proxy within the client’s Intranet. The results for “ABO Proxy” and “CGI Proxy” are obtained by connecting through our ABOProxy implementation and the CGIPProxy implementation, respectively. The results are normalized to make the measurements for “conventional web proxy”\(^5\) to be 1.

Compared with the conventional web proxy, the overheads in page load time and page size are introduced by the link rewriting performed by the proxies. In terms of the page load time, the ABOProxy has 47% overhead on average, while the CGIPProxy has 620% overhead. In other words, the link rewriting performed by the ABOProxy is 13 times faster than that of the CGIPProxy on average. The link rewriting in the ABOProxy merely appends .aboproxy.com to the end of all hostnames using pattern matching. The CGIPProxy has to parse all HTML and JavaScript documents to find all links and modify them accordingly, where the logic is quite complicated and time-consuming.

In terms of the page size, the ABOProxy has 2.6% overhead on average, while the CGIPProxy has 27% overhead. The ABOProxy merely appends .aboproxy.com to the end of all hostnames in the server’s response. The CGIPProxy has to rewrite all links contained in the server’s response, including both relative links and absolute links. For dynamically generated links, the CGIPProxy needs to modify the JavaScript code and include its own JavaScript library if necessary.

#### 5.2.2 Server throughput and responsiveness

The server scalability is typically estimated by measuring the responsiveness and throughput of the server when gradually increasing the number of concurrent clients. The server responsiveness is measured by the average Time To First Byte (TTFB) on the clients. The TTFB is the duration from the client making an HTTP request to the first byte of the page being received by the client. The server throughput is measured by the maximum number of requests that are processed by the server every second. We simulated the concurrent clients using the Web Application Stress Tool from Microsoft. The evaluation results are shown in the Figure 5.

The server throughput for the ABOProxy is 12.88 requests/sec, while the throughput for the CGIPProxy is only 3.25 requests/sec. The throughput of the ABOProxy is about 4 times larger than that of the CGIPProxy. We call the concurrency level beyond which the throughput does not increase any more as the server’s capacity. Before the concurrency level reach the server’s capacity, the TTFB grows very slowly. After the concurrency level exceeds the server’s capacity, the TTFB grows linearly to the concurrency level. In real world system, people try to avoid the number of concurrent users exceeds the server’s capacity by overestimating the server loads and deploying more powerful server machines. As shown in Figure 5, in all situations, the TTFB of the ABOProxy is 4-6 times less than that of the CGIPProxy.

### 5.3 Security: Same-Origin Policy

\(^5\)As stated in the website of CGIPProxy, though this is a beta release, it is stabler than the old production release.
Table 4: Popular websites failed in the CGIProxy implementation.

<table>
<thead>
<tr>
<th>Websites</th>
<th>Failed operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gmail</td>
<td>Send an email; View/edit settings/contacts/labels</td>
</tr>
<tr>
<td>Microsoft Exchange Email</td>
<td>Every link broke after login</td>
</tr>
<tr>
<td>Facebook</td>
<td>Search friends; Add a friend; Confirm a friend request; Join a group; Send a message</td>
</tr>
<tr>
<td>Hotmail</td>
<td>Manage calendar; Manage contacts; Write a new message; Manage folders; Add a blog entry</td>
</tr>
<tr>
<td>Yahoo</td>
<td>Check emails; After login, when performing personal operations, it asked for a login again.</td>
</tr>
<tr>
<td>Expedia</td>
<td>Search for a flight</td>
</tr>
<tr>
<td>Youtube</td>
<td>Play a video</td>
</tr>
</tbody>
</table>

Figure 4: The performance evaluation on page load time and page size

Figure 5: The performance evaluation on the throughput and TTFB under heavy load
A fundamental security model applied by the modern browsers is the same-origin policy. The same-origin policy is enforced by isolating webpages according to the security context of a three-tuple <protocol, hostname, port>. For example, the script from http://www.evil.org cannot access the contents downloaded from https://bank.com.

In the design of the CGIProxy, the original URL is encoded in the path of the new URL. In other words, every destination domain is mapped to a directory in the host of www.cgiproxy.com. From the browser’s point of view, all websites the user visits through the proxy are hosted at http://www.cgiproxy.com (or https://www.cgiproxy.com if the proxy server is using HTTPS), so that they have the same security context and are allowed to access one another. As a result, a malicious website is able to launch cross-site scripting attacks and cookie-stealing attacks against a sensitive website given that the user visits both through the proxy. In conclusion, the design of the CGIProxy completely destroys the domain boundary enforced by the same-origin policy in the browser.

In contrast, in ABOProxy, each destination domain is mapped to a subdomain of the proxy. From the browser’s point of view, different websites the user visit through the proxy are hosted in different hosts, so that they have different security contexts and are isolated by the same-origin policy. Therefore, our approach preserves the domain boundary among different websites and does not introduce any security concerns.

6. APPLICATIONS AND EXTENSIONS

A Search Cache without Broken Links

There are several services which archive Internet content to allow retrieval of pages as they looked at different points in time. The service web.archive.org is probably the best known. This makes available snapshots of the web at different points in time. This archive effectively acts as a link-translating proxy between the user’s browser and an archived copy of the web. For example, the NY Times from Jan 8, 2001 is available at: web.archive.org/web/20010108094000/http://www.nytimes.com/. As can be seen, the target URL is encoded in the path. Thus the archiver must find and translate all absolute and relative links, just as CGIProxy does. Rather than store things this way we can encode the URL in the host. For example, if we used the format .DD.MM.YYYY.ABOarchive.com then the above page becomes www.nytimes.08.01.2001.ABOarchive.com. This has the advantage that we can evade the hunt for relative links.

Each of the major search engines offers cached versions of pages. This can be useful when the server is off-line or the content sought is no longer available. This cache effectively acts as a link-translating proxy between the user’s browser and an archived copy of the web. For example, Table 5 shows the URL for the cached version of the page http://en.wikipedia.org/wiki/Pacific_Madrone in the Google, Yahoo! and Bing search caches. In each case the target URL is encoded in the path. Links within the cached page are to be served from the origin server, not the cache. For example, the link to the wikipedia page for “arbutus” should be served from wikipedia, not from the Google cache. Thus, when storing pages in the cache absolute links are broken, but relative links must be found and converted to absolute. As we have seen, this is an error-prone-process; thus broken links are common when viewing pages from the search caches.

Rather than change these documents in the cache, we propose letting the browser do the hard work for us. Suppose we crawl a document www.foo.com/doc1. If we serve this to users from cc.msnocache.com we must find and change links. But suppose instead we serve it from www.foo.com.ABOcache.com/doc1. E.g. suppose there’s a relative link to document /dir/doc2. We may or may not have this document in the search cache, but in either case we desire that it be served from the origin server. For this we do as follows:


2. If Request Header Referrer contains ".ABOcache.com" then "Return 301 document moved to: www.foo.com/dir/doc2 (just strip the .ABOcache.com fields") " Else serve the document from the cache


The else clause in step #2 ensures that documents that are the result of a user requesting a cached document are distinguished from those that are links in cached documents. Thus, users who request from the cache get the cached document, but all links therein are served from the origin server.

The advantages of this approach are twofold. First, no effort is required to translate any links. This system is a simplification of ABOProxy in that, not even absolute links need be found. Second, broken links in the cache are almost eliminated: they occur only if they have disappeared from the origin server. Thus the broken links will be strictly fewer than using current approaches, which break either if a relative link is mis-mediated or a document disappears.

A Simple Parasitic Web-Crawler Crawling has become core technology for those doing web research. This has given rise to need for efficient crawlers, which support these applications. Generally these work by starting at a list of seed documents which are retrieved and
parsed for web links. All links found are added to a queue. Links are then taken from the queue and the corresponding documents retrieved in turn. These documents are themselves parsed for links and so on. Most crawlers thus contain the following components: a crawl frontier which stores the list of URLs to be downloaded, a name resolver which turns names into IP addresses, a downloader which retrieves documents using HTTP, a link extractor that finds links in downloaded documents, a duplicate detector, which determines if a link has been encountered before. A good crawler should be scalable, efficient and “nice” (i.e., the crawler does not direct burdensome traffic to the server).

Each of the components of a crawler can involve non-trivial engineering. For example, even a seemingly simple step such as DNS name resolution can result in a significant bottleneck that greatly limits crawl throughput [16]. Equally, determining that a link has been encountered before is non-trivial. A good design must protect against crawler traps. These are sets of URLs that trap the crawler in a crawling loop which never terminates, and thus uses resources for no benefit. Thus, constructing a scalable crawler that is matched to the hardware and bandwidth resources available, is a difficult task and the subject of ongoing research. Haydon and Najork [16] and Cho and Garcia-Molina [17] describe scalable extensible crawlers.

We now show how ABOProxy may be used to avoid these design issues and build a Simple Parasitic Crawler (SPCrawler). Rather than solve each of the problems of design directly SPCrawler relies on the fact that others have already solved them, and leverages their solutions. SPC entirely eliminates this concern: niceness, policy and rate-limiting are done by the host crawler. This works as follows. SPCrawler is simply an instance of ABOProxy which retains copies of all of the documents fetched. To begin the crawl we first advertise a seed link where it will be found by a host crawler. For example, if we wish to seed a crawl with www.nytimes.com we would advertise a link to www.nytimes.com.spcrawler.com where the host crawler (e.g., msnbot) will find it. On retrieving the link msnbot’s link extractor will find any relative or absolute links on the page and add them to its crawl frontier. Of course the link travel.nytimes.com.spcrawler.com to msnbot and so on. Thus, all links on the seed page will look like travel.nytimes.com.spcrawler.com and so on. Thus, all links on the seed site will again be fetched through SPCrawler and so on. In this way the host crawler is induced to fetch more and more documents, always passing through SPCrawler as shown in Figure 6. The complex design questions of crawler design do not have to addressed, as they are handled by the host.

We can request all but the desired crawler stay away. The following robots.txt requests that msnbot retrieve any content except from the private directory and that all other crawlers stay away.

```
User-agent: msnbot
Disallow: /private
User-agent: *
Disallow: / 
```

By default search engine crawlers index content which then becomes available to their users in the form of search results. Using SPCrawler a second copy will be created. For example, the page www.foo.com.spcrawler.com/diri/doc1 will be indexed and stored in the cache, but the original www.foo.com/diri/doc1 will also be there. These two pages will be identical (except for absolute links which have .spcrawler ap-
We extended the ABOProxy implementation as an anonymous proxy. Besides normal anonymization steps, such as removing third-party cookies and web bugs, the key step is to encrypt the communication between the client and the proxy server using HTTPS. With ABOProxy, the protocol between the client and the proxy is the same as that between the proxy and the website. To work around this restriction, we extend the link rewriting of ABOProxy by modifying all appearances of \texttt{http://} in the server’s response to \texttt{https://non-secure}. In this way, all communication between the client and the proxy is over HTTPS, and the proxy is able to tell whether the original URL is using HTTP or HTTPS by examining the first filed of the hostname. If the first field of the hostname is \texttt{non-secure}, the proxy changes the protocol to HTTP and remove the first filed of the hostname.

**WebVPN.** We built a WebVPN service using the ABOProxy implementation to enable the employees to access the internal enterprise network from outside locations. The enterprise network is using the Microsoft Active Directory Domain Services. Each internal website is an active directory, such as \texttt{team, paystub, hrweb, etc..} The VPN server machine connects to the internal network, and at the same time has an external IP address that is accessible from the Internet. The communication between the clients and the VPN server is using HTTPS. As a result, the active directory \texttt{hrweb} can be accessed using the WebVPN at \texttt{https://hrweb.aboproxy.com}. A single wildcard SSL certificate for \texttt{*aboproxy.com} is enough to cover arbitrary active directories. The VPN server rejects the requests targeting a website that does not exist in the internal network.

**One-time-password System.** We have integrated the ABOProxy implementation with the one-time-password (OTP) system proposed in [18]. The system enables the user to safely log into sensitive websites in public computers without worrying about the keyloggers.

The user carries a list of one-time codes, each of which is an encrypted version of the password. When logging in the user delivers the encrypted password to the proxy, which decrypts and forwards to the end server. The proxy effectively stays as a man-in-the-middle for the session. The system does not require changes to existing authentication servers, and is running at \texttt{www.urrsa.com}.

7. CONCLUSION

The key challenge for a link-translating proxy lies in the link rewriting process. This paper proposes a novel architecture that uses a sub-domain mapping technique that makes the link rewriting simple and robust. We implemented ABOProxy using IIS and ASP.NET, and tested it by using it for everyday browsing. Both the theoretical analysis and experimental evaluation show that the ABOProxy is significantly better than existing web-service based proxies and webVPN’s in terms of robustness, performance, and security.

8. REFERENCES

About the authors:

Cormac Herley is a Principal Researcher at Microsoft Research. His main current interests are data and signal analysis problems that reduce complexity and help users avoid harm. He's been at MSR since 1999, and before that was at HP where he headed the company’s currency anti-counterfeiting efforts. He received the PhD degree from Columbia University, the MSEE from Georgia Tech, and the BE(Elect) from the National University of Ireland. He has authored more than 50 peer reviewed papers, is inventor of 70 or so US patents (issued or pending) and has shipped technologies used by millions of users.

Ziqing Mao is a Software Engineer at Facebook. His main interests are system security and web security. His research aims at providing a more reliable, secure and efficient web browsing experience by improving the browser design, hardening the operating system and improving the web service infrastructure. Before joining Facebook, he received the PhD degree from Purdue University in 2009.
Coordinating e-Health Systems with TuCSoN Semantic Tuple Centres

Elena Nardini Andrea Omicini Mirko Viroli
ALMA MATER STUDIORUM – Università di Bologna, Cesena, Italy
{elena.nardini, andrea.omicini, mirko.viroli}@unibo.it

Michael I. Schumacher
University of Applied Sciences Western Switzerland (HES-SO), Sierre & Fribourg, Switzerland
michael.schumacher@hevs.ch

ABSTRACT
Open and distributed application scenarios like e-Health systems mandate for new coordination models and technologies. In particular, they require middleware providing coordination and security services modelled with abstractions promoting run-time observability and adaptation.

Along this line, in this paper we describe the architecture of the TuCSoN coordination infrastructure, and show its application to an e-Health application scenario.

1. INTRODUCTION
Healthcare supported by software systems – in short, e-Health – is evolving quickly [7, 24]. Among the several e-Health research activities, research on Electronic Health Record (EHR) is particularly intensive [24, 43]. An EHR represents a set of medical information called fragments, which are stored in a digital format over different healthcare institutions. The introduction of EHR offers several benefits [26]: better patient safety, lower cost of health services, and improvements in healthcare audit and research. The main challenge in the EHR domain is to ensure interoperability among EHR fragments belonging to an environment that is distributed and open, and where the security support represents a fundamental requirement to protect the patient privacy [23]. Several efforts have been made in the EHR domain in order to cope with such requirements, in particular:
- Definition of standards for EHR-fragment format and communication such as Health Level Seven (HL7) [11] and Digital Imaging and Communications in Medicine (DICOM) [21].
- Definition of specifications like the IHE (Integrating the Healthcare Enterprise) specifications [20], to build EHR coordination middleware able to coordinate EHR-fragment providers and requesters.
- Definition of EHR coordination middleware based on the semantic tuple-space computing [5], such as the one promoted by Triple Space Computing (TSC) [1].

As shown by TSC, semantic tuple-space computing provides models for EHR coordination middleware which overcome the limits of the solutions proposed in the IHE specifications. In fact, unlike IHE, TSC provides a solution based on semantic techniques as suggested by openEHR and CEN EN 13606, and models for building applications that support more complex dynamics than the mere exchange of EHR fragments. On the other hand, like IHE, TSC provides coordination media that cannot be tailored to the specific application needs. Thus, any coordination law not directly supported by the model has typically to be charged upon coordinated entities, thus increasing their complexity, especially in open scenarios. Moreover, TSC cannot cope with the evolution of applications over time, since it makes it difficult to adapt the coordination middleware in case of changes to application requirements.

Along this line, in this paper we aim at providing models and approaches extending the solutions proposed in IHE and TSC, in order to augment their effectiveness in building EHR services, in particular as far as interoperability is concerned. To this end, we draw our inspiration from TuCSoN [33], an open-source coordination infrastructure based on semantic tuple-centres [27], and promoting online engineering [35] through a middleware adaptable at runtime, thus coping with the dynamism requirements.

In particular, this article is organised as follows. In Section 2 we briefly survey the existing approaches supporting interoperability among EHR fragments, and discuss their benefits and drawbacks. Then, in Section 3 we present an overview of the TuCSoN architecture. In particular, we describe the key-features of the architecture: (i) the behaviour programmability and the semantic support of tuple centres— the coordination abstraction exploited by TuCSoN; (ii) the organisation and RBAC models, used respectively to model the system structures and their relationships, and the security aspects; and (iii) the online engineering approach, used to support the corrective/adaptive/evolutive maintenance of software systems. In Section 4, we show how to exploit the key-features of the TuCSoN architecture in order to extend the solutions proposed in IHE and TSC. Finally, Section 5 concludes, providing final remarks.
2. EHR SYSTEMS INTEROPERABILITY

2.1 Towards electronic health records

The healthcare domain has been evolving quickly over the past decades. Nevertheless, advances are somewhat limited in several domain [3] and obstacles are still to be overcome [10, 36]. Most of the administrative processes have adopted an e-Health solution so as to become computerised. However, in some hospitals and for the large majority of General Practitioners (GPs), medical data is still acquired and exchanged on paper. The totally paperless hospital has yet a way to go [38]. Besides digital storage, the computerised acquisition of medical data also makes data accessible for computerised decision support [42] and has the potential to reduce the large number of adverse events, particularly in hospitals [7, 24]. Alongside local advantages, communication of health data is another important factor in computerised data acquisition to overcome limits of paper-based information exchange, which is often slow [41] and error prone [24]. e-Health information exchange strategies and solutions have been developed on a local regional or cross-institutional [25] and national [34, 14] level and some already have concrete implementation in research projects [41].

Among the several e-Health research activities concerning the health information exchange, research on Electronic Health Record (EHR) is particularly intensive [24, 43]. A Patient EHR-document refers to the medical record of a patient stored in a digital format. The information stored in an EHR might include patient information such as demographics, medical history, medication, allergy list, lab results or radiology. Medical data belonging to an EHR are called fragments, and can be distributed over different EHR systems. The introduction of EHR offers several benefits [26]:

Better patient safety — Storing and transferring patient information electronically allows reducing clinical errors caused for example by illegible handwriting, documents or images, as well as it allows clinicians to communicate more quickly and accurately and to identify relevant information more easily.

Lower cost of health services — EHR technology can reduce administrative work to manage medical data since it can increase medical-data search efficiency and reduce medical-data duplication and waste.

Better audit and research — Behind improving medical assistance of patients, EHR technologies are also useful for other purposes. In particular, electronic databases of health information can be exploited for healthcare audit and research.

In order to keep the EHR benefits, EHR systems should ensure interoperability among EHR fragments. Interoperability — that is, the ability for two or more heterogeneous systems to communicate together — is of paramount importance in health information communication [12]. In case of EHR systems, interoperability should satisfy the following conditions [23]:

Distribution — EHR fragments should be easy to share even if the information is widespread across multiple EHR systems.

Openness — EHR supporting servers at different caregivers could be heterogeneous and change dynamically.

Security — It is necessary to support security mechanisms in order to avoid failures that can cause injury to the patient and violations to privacy.

Accordingly, interoperability among EHR systems call for specialisation middleware able to deal with distribution, openness and security requirements in a coherent and transparent way. In the next section, we discuss standards and solutions from the literature, which propose design principles for middleware of such a sort.

2.2 Existing approaches: a survey

In order to cope with distribution, openness and security, the first approach to the issue of interoperability is the definition of standards for EHR-fragment format and communication. The two most representative are [23]:


- **Digital Imaging and Communications in Medicine (DICOM):** a standard for handling and transmitting information in medical imaging. It includes a file format definition and a network communication protocol [21].

Standards such as HL7 and DICOM are not enough to achieve interoperable health systems. In fact, the result is that EHR systems use different set of format and communication standards, often incompatible, incomplete or involving overlapping scopes, thus breaking the interoperability requirement [18]. As a response to these problems — and as a complementary step towards the requirements of interoperability among EHR fragments — the following standards and initiatives were proposed:

- **openEHR [13] and CEN EN 13606 [15]:** standards aiming at facing interoperability among EHR fragments. In particular they propose semantic approaches based on Archetype Definition Language (ADL) [4] — a formal language for expressing application-domain concepts — in order to describe semantically EHR fragments. By exploiting such kind of semantic techniques it may be possible to support interoperability among EHR fragments with different syntactic structures depending on the adopted standard.

- **Integrating the Healthcare Enterprise (IHE) [20]:** a non-profit initiative founded in 1998 led by professionals of the e-Health industry. The initiative goal is not to develop standards as such, but to select and recommend an appropriate usage of existing standards (e.g. HL7 and DICOM) in order to improve the sharing of information among EHR systems.

In this context, openEHR, CEN EN 13606, and IHE emphasise two important requirements. The first is to describe semantically EHR fragments in order to face heterogeneity and dynamism of fragment formats. The second is to provide a coordination middleware able to coordinate EHR systems and actors interacting with such systems, hiding distributed-fragment management and security issues from
entities to be coordinated. In particular, XDS, ATNA and XUA are central profiles for building a coordination middleware that connects EHR systems.

A further important contribute in this context is represented by Triple Space Computing (TSC) [5], which provides a different solution based on the Linda tuple space model [17]. Through the tuple space model, coordination among system entities occurs by exchanging information in form of tuples, in a common shared space called tuple space. System entities to be coordinated communicate with one another through out, rd and in primitives to respectively put, read and consume associatively tuples to/from the shared space. In particular, TSC shows the following interesting features:

- It provides a general coordination model to manage all kinds of interactions among system entities.
- Tuple space model is based on generative communication [17]: tuples generated by a tuple producer have an independent existence in the tuple space leading to time, space and name uncoupling. Uncoupling is a requirement to satisfy in order to cope with openness. Entities belonging to an open environment can be heterogeneous and can be added, removed or modified at runtime. For this reason, an entity cannot make a-priori assumptions about other system entities.
- It is based on semantic tuple-space computing [28]: it adopts tuple spaces enriched semantically thus allowing an exchange of data (tuples) semantically described by means of an ontology.
- Like IHE, it provides a Web-service interface to tuple spaces which promotes interoperability.

TSC puts together the advantages of using openEHR, CEN EN 13606 and IHE. Moreover, it improves the IHE approach by proposing a more general coordination model suitable for open scenarios, and not only specialised on the storage and retrieval of EHR fragments. In particular, through TSC it is possible to exploit a unique coordination model – the tuple space model – to manage all the system interactions. This is useful in case it is needed to extend e-Health systems with coordination functionalities concerning different kinds of interactions. For example, interactions with patients, with scientific research systems or with systems providing consumers with access to medical developments and research.

However, TSC exhibits some limits, too. The first one derives from the Linda tuple-space model: the tuple space behaviour is set once and for all by the model and cannot be tailored to the specific application needs [31]. Thus, any coordination law not directly supported by the model has typically to be charged upon coordinated components, thus obstructing the way to open systems. A further feature that should be supported by an EHR coordination middleware – and that is not covered by either IHE or TSC – is the ability to change its configuration at runtime in order to cope with application dynamism. In fact, during the lifetime of an application, requirements could be changed, added or removed. For example, new nodes could be added/removed to/from the network, or, coordination algorithms could be changed in order to improve the efficiency and effectiveness of the overall application. If the middleware does not allow for runtime changes, it might be necessary to shut it down in order to update its configuration—which is definitely undesirable, especially in application scenarios like e-Health that require continuous service availability.

Figure 1: Logical levels for a coordination middleware

As far as middleware for the health domain is concerned, the current state of the art can be summarised as follows:

- IHE profiles, in particular XDS profile, do not provide a middleware model expressive enough to manage interactions among EHR actors. In particular, XDS provides a coordination middleware model not based on semantic techniques, and focused on coordinating meta-data in order to store and retrieve EHR fragments. As a consequence, it cannot be used for e-health applications going beyond the mere fragment coordination.
- TSC provides a solution that overcomes part of the XDS profile drawbacks. In particular, it exploits the Linda tuple-space model enriched with semantic techniques that exhibits features particularly useful for the realisation of an EHR middleware. On the other hand, TSC has some limits due to the fixed behaviour of its coordination abstractions, and to its inability to cope with middleware evolution over time.

Accordingly, an EHR middleware should be a coordination middleware supporting interactions among heterogeneous EHR-fragment providers and requesters—as in Figure 1. The middleware should be developed upon a coordination infrastructure giving the support for distribution of heterogeneous EHR-fragment-stores and providers/requesters of EHR fragments in a transparent way. Then, the infrastructure should provide the API required to build a coordination and a security service. The coordination service has the task to enable and rule interactions among system actors. It should exploit a general-purpose coordination model based on the tuple-space model and on semantic techniques, in order to cope with the openness requirement. In turn, the security service should be able to guarantee privacy of patient EHR-fragments, taking into account the federated nature of the healthcare system. Finally, the coordination infrastructure
should be based on engineering approaches making it possible to build a coordination middleware whose configuration is adaptable at runtime, in order to maintain a continuously-available EHR-service in front of dynamic changes of the application requirements.

3. THE TuCSoN ARCHITECTURE

TuCSoN (Tuple Centres Spread over the Network) [33] is a coordination model & infrastructure that manages the interaction space of a distributed software system by means of tuple centres [31]. Tuple centres are tuple spaces [16] enhanced with the ability to define their own behaviour in response to interaction events, according to specific application needs. In the same way as tuple spaces, tuple centres are information spaces structured as sets of tuples, i.e., structured and ordered chunks of information data; system components can interact with and through tuple centres by inserting, reading, and consuming tuples—so, the tuple centre model provides the same benefits of the generative communication as tuple spaces [16], which leads to system components uncoupling in space, time, and components’ name. From the topology point of view, tuple centres are distributed and hosted in TuCSoN nodes, defining the TuCSoN coordination space [6]. In particular, the topological model of TuCSoN classifies nodes as places and gateways—as shown in Figure 2. The former represent the nodes hosting tuple centres used for specific applications/systems need, from supporting coordination activities to hosting information or simply enabling software components communication. The latter provide instead information about a the set of places belonging to a single domain—thus avoiding a single and centralised repository, which is unfeasible in complex and large environments. A domain is the set of nodes composed by the gateway and the places for which it provides information. A place can be part of different domains and a gateway can be a place in its turn. The overall picture of the TuCSoN topology is provided in Figure 2.

Besides topology, the key features of the TuCSoN architecture are: (i) the behaviour programmability and the semantic support in tuple centres, that represent the coordination model supported by TuCSoN—those aspects will be treated in Section 3.1; (ii) the use of the organisation and RBAC models: the former is exploited to describe the system structures and their relationships, the latter is exploited to model security aspects—both models will be treated in Section 3.2; and (iii) the online engineering approach used to support the corrective/adaptive/evolutive maintenance of software systems. Such an approach will be treated in Section 3.3.

3.1 Behaviour & semantics in tuple centres

The behaviour of the original tuple spaces—represented by their state transition in response to the invocation of the standard coordination primitives—is set once and for all by the model, and cannot be tailored to the specific application needs [16]. As a consequence, any coordination policy not directly supported by the standard behaviour of the coordination abstraction—the tuple space—has to be charged upon system coordinables, which hence grow in complexity—thus hampering the effectiveness of the coordination model especially in open scenarios [33]. Moreover, associative access [16] to tuples—tuples are retrieved by content and not by reference—in standard tuple spaces is based on a tuple matching mechanism which is purely syntactic. Although this might appear as a marginal aspect of tuple-based coordination models, it however imposes to coordinated components a design-time awareness of the structure and content of tuples: ultimately, components coordinated through a tuple space should be designed altogether—thus clearly working against the basic requirements for openness [40].

In order to overcome the above limits, the semantic tuple centre model adopts respectively behaviour programmability and semantic support. Through the behaviour programmability it is possible to program the tuple centre behaviour so as to embed coordination policies within the coordination media, without charging upon system coordinables. Whereas, through the semantic support it possible to exploit ontology languages to semantically describe information in tuple centres. Thus, the associative access provided by tuple centres could exploit a semantic content description without requiring a design-time awareness of the structure and content of tuples, but only of the ontology describing the application domain.

3.1.1 Behaviour programmability

The tuple centres behaviour can be determined through a behaviour specification, defining how a tuple centre should react to incoming/outgoing coordination events [31]. The behaviour specification can be expressed in terms of a reaction specification language that associates any communication event possibly occurring in the tuple centre, to a set of computational activities called reactions. Each reaction can access and modify the current tuple centre state by adding or removing tuples and access all the information related to the triggering communication event such as the performing software component, the operation required and the tuple involved. So, differently from tuple spaces, tuple centres represents general-purpose and customisable coordination media that can be programmed with reactions tailored to the application needs.

TuCSoN exploits ReSpecT tuple centres [31, 30]. ReSpecT adopts a tuple language based on first-order logic, where a tuple is a logic fact, any unitary clause is an admissible tuple and reactions are defined through logic tuples, too. A specification tuple is of kind reaction(E, G, R). It associates a communication event described through E, to the reaction R. G represents a set of conditions that has to be satisfied in order to execute a reaction R, if the incoming event matches E. A reaction is defined as a transactional sequence of re-
Semantic tuple templates.

Semantic templates are to be used to flexibly retrieve semantic tuples, hence they consist in specifications of set of domain individuals described by the domain ontology. As for semantic tuple, in ReSpecT tuple centres semantic tuple templates are logic tuples. In [27] was also defined a SHOIN(D)-like language to represent semantic logic templates like

‘Car’(exists hasMaker : ford)

which describes the set of individuals of kind Car in hasMaker-relation with the individual ford.

Semantic reactions.

As shown in Section 3.1.1, a ReSpecT specification reaction is a tuple of kind reaction(E,G,R). In a semantic specification reaction, E describes the kind of primitive which it refers and a semantic tuple template. For example, E could be something like out(Car) describing every out of a semantic tuple representing an individual that has to belong to the concept Car. Otherwise, E could be something like in(Car) referring to every in with a semantic template describing a concepts that has to be sub-concept of Car.

Then, the guard G represents a set of conditions that has to be satisfied in order to execute a reaction R. Thus, in a semantic reaction specification it may contain conditions about the semantic tuples and templates described in E. In other words, G can contain concept descriptions that are to be respected by tuples or templates unifying in E.

Finally, since R can contain the use of the primitives in, out and rd, it can semantically interact with the tuple centre, as well as every system component.

Semantic matching mechanism.

In case of primitives, semantic matching simply amounts at checking whether an individual, in form of a semantic tuple, is an instance of the concept described by a semantic template. Considering semantic reactions, in case of out, semantic matching amounts at checking whether the individual described through the primitive is an instance of the individual-set specification described in E and G. Whereas, in case of in or rd, semantic matching means checking whether the concept described through the primitive is a sub-concept of the concept described E and G.

Like every DL-based system, SHOIN(D)-based systems provide a reasoner offering a set of reasoning services [2]. In order to implement the semantic matching mechanism for ReSpecT tuple centres the subsumption checking and instance retrieval services of the Pellet OWL-DL reasoner [39] were exploited. The former check if a concept A is a sub-concept of a concept B. Whereas, the latter service retrieve a set of individuals belonging to a given concept A.

Semantic primitives.

Semantic primitives (in, out and rd) represent the language whereby system components can read, consume and write knowledge described by means of a domain ontology. Since the knowledge stored in the semantic tuple centre must be always consistent with the domain ontology, in face of the primitive out it is required to check – by exploiting the DL reasoner – the consistency of the semantic tuple to be written in the tuple centre, with the domain ontology. This means that, differently from the original tuple centre semantic, the primitive out performed with a semantic tuple can
fail in case the related semantic tuple is not consistent with the domain ontology.

According to the above description, ReSpecT semantic tuple centres provide two main advantages. First, they exploit SHOIN(D)-like languages, in particular OWL-DL as their ontology language. OWL-DL represents a good compromise between expressiveness and complexity [19], and fits well the openness requirement, since it is a standard ontology language introduced for the Semantic Web, by the W3C Consortium. Moreover, while other approaches have explored how to use tuple-based models in semantic-aware contexts [40], the approach adopted in [27] for semantic tuple centres aims at smoothly extending the standard tuple centre setting so as to fully exploit the power of DL in coordination of system components, independently from the application context. In this way, all the benefits of using ReSpecT tuple centres (see Section 3.1.1) are maintained along with the semantic support.

3.2 Organisation & security

Jennings [22] refers to organisation as a tool helping software engineers managing complexity of software system development. Organisation allows the interrelationships between the various components of the system to be defined and managed, by specifying and enacting organisational relationships. In this way, engineers can group basic components into higher-level unit of analysis, and suitably describe the high-level relationship between the units themselves.

TuCSoN exploits an organisational model that is role-based [32]. Organisation and coordination are strictly related and interdependent issues. Organisation mainly concerns the structure and the structural relations of a system — i.e., the static issues of the agent interaction space. Coordination mainly concerns the processes inside a system — i.e., the dynamic issues of the agent interaction space —, often related to roles that usually frame agents position in the structure of the system organisation. Moreover, coordination is strictly related to security, being both focused on the government of interactions inside a system, however according to two different (dual) viewpoints: normative for security, constructive for coordination [6]. Whereas security focuses on preventing undesired/incorrect system behaviours — which may result in problems like denial of services or unauthorised access to resources — coordination is concerned with enabling desirable/correct system behaviours, typically the meaningful, goal-directed interaction between different system components. Due the relations among coordination, organisation and security, TuCSoN exploits an unique and coherent conceptual framework to manage the complexity of modelling such three dimensions [32].

The TuCSoN conceptual framework is represented by an extended version of the Role-Based Access Control (RBAC) model [37] — as shown in Figure 3 — called RBAC-MAS [44]. The model interprets an organisation as a set of societies composed by software components playing certain roles according to the organisation rules, where each role has an associated set of policies. Organisation rules define two types of relationships among roles: (i) component-role relationship, through which it is possible to specify whether a specific component is allowed (or forbidden) to assume and then activate a specific role inside the organisation; (ii) role-role relationship, through which it is possible to specify structural dependencies among roles, so as to further define constraints on dynamic component role-activation.

A policy represents an admissible interaction protocol between the associated role and the rest of the organisation. An ACC (Agent Coordination Context) [29]) represents an entity contracted by a component, on the basis of its identity, when it enters the organisation. The ACC is then released to components and used from components in order to interact with the resources (here, the tuple centres) belonging to a specific organisation. The interaction is enabled and ruled by the ACC in accordance with the rules and policies defined by the organisation.

From a topology point of view, an organisation is mapped onto a domain (including the linked domains or sub-domains). The description of the structures and rules characterising the organisation are stored and managed dynamically in a specific tuple centre, called $ORG(OrgID)$ — where $OrgID$ is the organisation identifier —, hosted in a gateway node of the domain. The $ORG$ tuple centres host then information about societies, roles, components and the related relationships defined for the domain, represented by the gateway and its places.

3.3 Online engineering

The openness of software systems calls for keeping the abstractions alive [35]. Alive abstractions are defined in an explicit way in the meta-model of the system-engineering paradigm. Moreover, they are “kept alive” through the whole engineering process of a software system — from the analysis to the corrective/adaptive/evolutive maintenance phase. Such abstractions enable the inspection of their current state at runtime, so as to allow dynamic monitoring of system components that they model, and their creation and modification, so as to allow a dynamic evolution of system components. By exploiting such kind of abstractions, software engineers are enabled to perform online engineering [35], that is, the capability of supporting system design, development and test, debugging and evolution while the system is running.

Tuple centres are modelled and built as alive abstractions. Accordingly, TuCSoN allows the runtime maintenance of both coordination laws and organisation structure and rules. In particular, it is possible to maintain and evolve the coordination laws at runtime by inspecting and creating tuple centres, and by modifying their state or behaviour.
Then, it is possible to maintain and evolve the organisation model since the organisation structure and rules are reified as knowledge encapsulated in the tuple centre $\Omega_R$.

By means of its “alive abstractions”, TuCSoN allows in principle both humans and (intelligent) software components to maintain and develop a software system. In order to support humans, TuCSoN provides the Inspector tool [9] enabling software engineers to first design and then observe and act on system structures and processes at runtime, working upon abstractions adopted and exploited for the design of a system. Besides, TuCSoN also provides intelligent agents with the API needed to create, inspect and modify tuple centres. In particular, since TuCSoN exploits ReSpecT tuple centres – which are logic tuple centres – it is possible to exploit agents capable of symbolic reasoning in order to autonomously maintain the structures.

4. EXPLOITING TuCSoN IN E-HEALTH

In the following, we show how the TuCSoN approach can be adopted in order to extend the solutions proposed by TSC and IHE (see Section 2): the overall goal is to increase the effectiveness of TSC and IHE approaches in coordinating EHR fragments. In particular, we discuss how such approaches can be integrated and extended with the key features of TuCSoN architecture, presented in Section 3. When dealing with IHE, we refer in particular to the following recommendations [20]: Cross-Enterprise Document Sharing (XDS) – i.e. profile describing an infrastructure for storing and registering medical documents –, Audit Trail and Node Authentication (ATNA) – i.e. profile describing security procedures – and Cross-Enterprise User Assertion Profile (XUA) – i.e. profile describing means to communicate claims about the identity of an authenticated principal (user, application, system,...) in operations that cross healthcare-enterprise boundaries.

TuCSoN topology and XDS Affinity Domains.

The e-Health environment is federated, that is, each healthcare enterprise belongs to a domain with other healthcare enterprises, using a common set of policies ruling interactions with and within a domain, and sharing common clinical documents. XDS calls each domain Affinity Domain. According to Section 3, the hierarchical topology of TuCSoN fits well with the sort of topology required by the EHR scenario. In particular, an Affinity Domain could be mapped in a TuCSoN domain whose gateway maintains the information about the policies and the structures associated to the domain itself. Then, each healthcare enterprise belongs to an Affinity Domain can be mapped in a TuCSoN place.

TuCSoN semantic tuple centres as fragment coordination media.

XDS provides a model to store and retrieve EHR fragments. Figure 4 shows the actor model defined by XDS. In particular the model is composed by:

**Document Source** | A healthcare point of service where clinical data is collected.

**Document Consumer** | A service application where care is given and information is requested.

**Document Registry** | A system storing ebXML descriptions of the clinical fragments to rapidly find them back.

![Figure 4: Actors for the IHE XDS profile](image)

**Document Repository** | A system that stores documents and forwards the metadata to the document registry.

**Patient Identity Source** | A system that manage patients and identifiers for an Affinity Domain.

The XDS actor model has two main drawbacks. Document Registry is exploited to store and search metadata describing EHR fragments whereby it is possible to retrieve the related document from the Document Repository. In particular, XDS suggests to realise the registry through the ebXML Registry standards.

However, the main limit of an ebXML Registry is that it describes metadata in XML, and retrieves metadata in face of a query written in XML and SQL format. This kind of knowledge representation and retrieval lacks the expressive power provided by semantic approaches exploiting ontologies. In fact, unlike an ontology, an XML schema does not allow the description of complex taxonomies among concepts like those exploiting subsumption relationships. Also, XML tools does not perform powerful reasoning over metadata like semantic reasoning, which is able instead to infer new knowledge that is not declared in an explicit way. Thus, ontology-based approaches are more suitable for engineering knowledge in open context where the knowledge structure can evolve and where software components only have a partial awareness about the overall knowledge.

Another limit of the ebXML Registry is that it promotes a pre-defined behaviour only able to store and retrieve metadata. As a consequence, in order to extend the behaviour of the registry, a layer should be upon it that would enrich the operational semantic behind its interface in order to implement the new desired behaviour. This, of course, would definitely augment the complexity of the system. In order to cope with complexity, instead, it would be desirable to be able to define new behaviours directly in the registry, customising the registry with the policies associated to a particular healthcare domain. For example, a policy would allow the registry to be distributed over different nodes belonging to the domain, instead of having an unique registry per domain, as suggested by XDS. By exploiting behaviour programmability of the coordination media, it would be possible to coordinate a set of domain registries collaborating with one another in order to search and distribute metadata within the domain, in a smart way.

This is why the tuple centre model looks like a good candidate to build a Document Registry. On one hand, a semantic tuple centre supports the semantic representation of
the stored knowledge – like TSC –, but – unlike TSC – it also provides a tuple/template language that is independent from the technology exploited to implement the semantic support. Thus, each domain can choose to exploit a particular semantic technology guaranteeing interoperability with other domains. On the other hand, since the behaviour of a tuple centre is programmable, it is possible to tailor the registry to specific application needs. Moreover, by exploiting logic tuple centres like ReSpecT tuple centres, it is possible to promote cognitive processes by exploiting rational agents.

**Exploiting TuCSoN RBAC model.**

As shown in Section 3, TuCSoN provides the organisation abstraction to describe the structures and rules composing a system. In particular, an organisation in an Affinity Domain could be mapped in the TuCSoN $ORG$ tuple centre managing the domain structures, like Document Registries and the domain places where e-Health enterprises are hosted, and defining the set of roles that can interact with the organisation along with a set of related policies to rule such interactions.

In the context of Affinity Domains, a role represents a class of identities that can interact with EHR fragments, whereas policies represent the admissible interactions for a specific role. Accordingly, the RBAC-MAS model [44] can be suitable integrated with security recommendations defined in ATNA and XUA. Such recommendations in particular require: (i) an authentication service able to authenticate users, (ii) access control policies, (iii) a secure communication between system actors, and (iv) a security service supporting cross-authentication among EHR domains. In order to satisfy such requirements, TuCSoN can be integrated with two technologies suggested by IHE: Kerberos authentication service, and Web Services as interface to access to TuCSoN organisations, so as to promote the interoperability requirement.

Thus, Web Services can be used to access to TuCSoN organisation in secure way by exploiting WS-Security, that is, a secure communication protocol developed by the OASIS-Open group. In particular, WS-Security includes both WS-SecureConversation – which can be exploited to ensure secure conversations among system actors –, and WS-Trust— which can be exploited to support cross-authentication among EHR domains. Through WS-Trust it is possible to establish trust relations among domains that are exploitable to accept requests coming from different domains without having to authenticate users again. Finally, by integrating the authentication service Kerberos with TuCSoN, user identities can be associated to roles and policies in the $ORG$ tuple centre, and be authenticated.

**Online engineering for continuos e-Health system interoperability.**

As discussed in Section 3, TuCSoN exploits active abstractions to model coordination, organisation and security, thus promoting their online engineering. By exploiting semantic tuple centres to model Document Registries and organisation to model the structures composing an Affinity Domain, TuCSoN makes it possible to support the runtime corrective/adaptive/evolutionary maintenance of an e-Health fragment system—that is, with no need to stop the system. This is particularly useful whenever application requirements are expected to change substantially over time. For instance, it may happen that new places hosting e-Health enterprises need to be added by reconfiguring Affinity Domains dynamically, or that roles and policies have to be added/removed/modified to cope with dynamic organisation changes. This would require to change the behaviour of a Document Registry to face the new application requirements. Through online engineering as supported by the TuCSoN architecture, the system could be evolved in a consistent way at runtime, maintaining a continuous interoperability among EHR systems. We think this is a crucial aspect to be considered in the engineering of e-Health applications, where a continuos service availability is indeed fundamental—and this is why we promote the integration of IHE recommendations within the TuCSoN architecture.

5. CONCLUSIONS

The most important requirement to be satisfied in EHR fragment coordination is the interoperability among healthcare systems in a scenario in which: (i) EHR fragments could be distributed among different healthcare systems, (ii) healthcare systems can be heterogeneous and change dynamically, and (iii) security mechanisms play a fundamental role to ensure patient privacy and safety. Several efforts can be found in the literature trying to cope with such requirements—like HL7, DICOM, CEN EN 13606, openEHR, IHE and TSC.

However, such approaches exhibit two main shortcomings. First of all, they provide special-purpose models of coordination, which increase the complexity of building an EHR coordination middleware. This limits system interoperability by making it difficult to integrate independent e-Health systems. Moreover, they do not support any form of online engineering. As a consequence, the coordination middleware cannot be updated at runtime in order to cope with new application requirements without stopping the system.

Along this line, in this paper we proposed a coordination model and technology that could integrate the solutions and standards proposed in literature while addressing the aforementioned issues. In particular, we proposed TuCSoN as a reference architecture for coordination in the e-Health scenario since it provides a general-purpose model of coordination accounting for distribution and security issues in the engineering of EHR systems, and promotes online engineering for continuos service availability of e-Health applications.

6. REFERENCES


About the authors:

Dr. Mirko Viroli is research associate at DEIS, the Department of Electronics, Informatics and Systems of the Alma Mater Studiorum–Università di Bologna (UNIBO), Italy. He is an expert in computation models, in the areas of object-oriented programming languages, agent-based systems, coordination infrastructures, and self-organising systems. He has written over 150 articles on such topics, of which more than 30 on international refereed journals. He currently leads the UNIBO team in the FP7 STREP project SAPERE (Self-Aware PERvasive Ecosystems).

Andrea Omicini is Full Professor at the Department of Electronics, Informatics and Systems of the Alma Mater Studiorum–Università di Bologna, Italy. He received his laurea degree in electronic engineering (1991) and his PhD in computer engineering (1995) both from the Alma Mater. He wrote over 200 articles on coordination, software infrastructures, multiagent systems, Internet technologies, software engineering and programming languages published in international journals, books, conferences and workshops. On the same subjects, he edited 15 books and guest-edited 10 international journal special issues. As an active member of the ACM SIGAPP, he chaired the ACM SAC conference (2004-2005) and some of its tracks (2000-2003, 2009-2010), working as a member of the Steering Committee (2000-2003); also, he was and currently is a member of the Editorial Board of the ACM Applied Computing Review (2000-2002, 2010-).

Prof. Michael Ignaz Schumacher is a full Professor in the Institute of Business Information Systems at the University of Applied Sciences Western Switzerland (HES-SO) since 2007. Previously, he was Research Associate at the Swiss Federal Institute of Technology Lausanne in Switzerland (EPFL) and visiting researcher at the Robotics Institute in Carnegie Mellon University in Pittsburgh (USA). He holds a PhD and a MsC in computer science from the University of Fribourg in Switzerland. His research focuses on distributed information systems and artificial intelligence applied to healthcare.