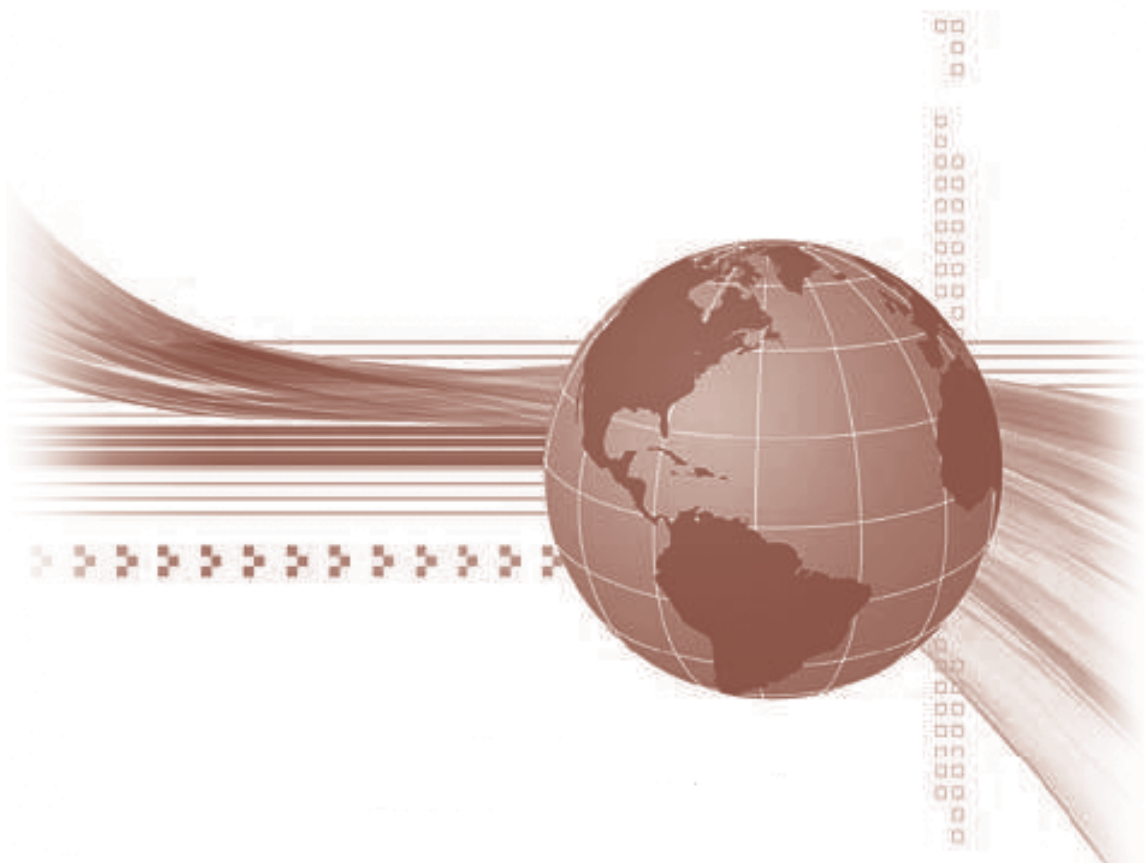
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

1/2015

STUDIA

UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

No. 1/2015

January - June

EDITORIAL BOARD

EDITOR-IN-CHIEF:

Prof. Militon FRENŢIU, Babeş-Bolyai University, Cluj-Napoca, România

EXECUTIVE EDITOR:

Prof. Horia F. POP, Babeş-Bolyai University, Cluj-Napoca, România

EDITORIAL BOARD:

Prof. Osei ADJEI, University of Luton, Great Britain
Prof. Florian M. BOIAN, Babeş-Bolyai University, Cluj-Napoca, România
Assoc. Prof. Sergiu CATARANCIUC, State University of Moldova, Chişinău, Moldova
Prof. Wei Ngan CHIN, School of Computing, National University of Singapore
Prof. Gabriela CZIBULA, Babeş-Bolyai University, Cluj-Napoca, România
Prof. Dan DUMITRESCU, Babeş-Bolyai University, Cluj-Napoca, România
Prof. Farshad FOTOUHI, Wayne State University, Detroit, United States
Prof. Zoltán HORVÁTH, Eötvös Loránd University, Budapest, Hungary
Assoc. Prof. Simona MOTOGNA, Babeş-Bolyai University, Cluj-Napoca, România
Prof. Roberto PAIANO, University of Lecce, Italy
Prof. Bazil PÂRV, Babeş-Bolyai University, Cluj-Napoca, România
Prof. Abdel-Badeeh M. SALEM, Ain Shams University, Cairo, Egypt
Assoc. Prof. Vasile Marian SCUTURICI, INSA de Lyon, France
Prof. Leon ŢÂMBULEA, Babeş-Bolyai University, Cluj-Napoca, România

YEAR **Volume 60 (LX) 2015**
MONTH **JUNE**
ISSUE **1**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

1

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

CREATING AN EFFICIENT AND INCREMENTAL IDE FOR TTCN-3

KRISTÓF SZABADOS

Abstract. In this article we present methods and algorithms for constructing an efficient IDE in the sense that the processing costs of reanalyzing source code after change is minimal. Moreover, we show that these methods and algorithms can be designed in a way that they support iterative realization, hence, they fit better to the current trends of iterative software development life-cycle. We also show how these algorithms can be built into an existing system and we show measurements on performance benefits. The proposed methods were validated in the telecommunication area for compiling TTCN-3 code.

1. Introduction

Nowadays developing a modern Integrated Development Environment (IDE) has two major requirements: (1) it has to be responsive and (2) developed in a lean iterative process.
(1) The common way of checking source code for errors is to have a compiler parse all of the files and run a semantic analysis on the produced data. In case of modern IDEs this is not feasible, full analysis takes too long on large projects. An IDE is expected to analyze the project automatically and to report possible errors instantaneously, irrespective of the size.
(2) The current software development trends favor iterative development in order to reduce cost. Instead of large development efforts supporting complex situations, the aim is to fullfill the needs of the user with a lean solution.

This paper is organized as follows. In Section 2 we present earlier works related to this subject. In Section 3 we introduce the datamodel and its

Received by the editors: April 25, 2014.

2010 *Mathematics Subject Classification.* 68N20, 68W40.

1998 *CR Categories and Descriptors.* D.2.3 [**Software Engineering**]: Coding Tools and Techniques – *Program editors*; D.2.6 [**Software Engineering**]: Programming Environments – *Integrated environments*; D.3.4 [**Programming Languages**]: Processors – *Incremental compilers*.

Key words and phrases. Parsing, Integrated Development Environment, Iterative Development.

operations. Section 4 describes the measurement environment. Section 5 presents the incremental parsing algorithm. Section 6 shows an outlook on how these methods can be used to speed up the semantic checking. Finally, Section 7 gives an analysis of the algorithm's performance, and Section 8 summarizes the results of this paper.

2. Related work

Several articles have already been published on incremental parsing. In programming languages recognition, languages are typically described by LL(k)[1], LL(*)[2], LR(0)[3], LR(1), and LALR(1)[4] grammars. Research on their incremental parsing focused mostly on LL [12, 7, 8] and LR parsing [3, 11, 5, 6, 10] methods. All techniques parse the input text, building the first parse tree, and update it according to the changes in the input. An important point in each case is finding the minimal structural units that are affected by the modifications ([3] for LR(0) and [6] for LR(k) grammars). Incremental parsers use "re-parse points" where parts of the semantic information could be re-parsed from the input. These occur close to the update location [7]. Others proposed approaches for doing incremental parsing with repairing errors coming from earlier processing as they get resolved later [1]. Wagner et al. [11] describes a method which does not use error recovery, but utilizes the history of modifications instead. There are also approaches applying neural networks (see [2] e.g. uses the Perceptron algorithm). Huang et al. [4] suggested dynamic programming to enhance incremental processing.

Most of these articles propose parsed syntax trees to enhance the re-parsing speed. Their aim is to minimize the parsing cost starting from any point in the tree. To reach this the authors produce parser generators which create incremental parsers from the Backus-Naur Form (BNF) of their target language. In this setup the tree is usually uniform and completely represents the given text in the file. The algorithms recommended by the authors work on the same way for each node, can modify any node in the tree and can generate full results in a single execution.

We propose a different approach. Instead of using a Parse Tree that represents an identical copy of the text in a tree shaped form, we use an Abstract Semantic Tree (AST). In an AST the semantic processing can change the

[1]LL is a top-down parser class for a subset of the context-free grammars. An LL parser is called an LL(k) parser if it uses k tokens of lookahead

[2]An LL parser is LL(*) if it uses the minimum lookahead per input sequence [9].

[3]LR is a bottom-up parser class, reading the input in one direction (typically left to right), producing a reversed rightmost derivation

[4]LALR parsers are simplified canonical LR parsers, with reduced language recognition power, and significantly reduced memory requirements

form and order of the information, for example order them alphanumerically, transform for efficiency or extend with external information if required. The iterative development approach allows us to implement a tool that generates the solution stepwise, i.e., it can be developed incrementally until the cost of further developments overweight its benefits.

We analyzed how the incremental parsing can be used to speed up semantic checking. A process missing from the articles mentioned earlier.

3. The model

In this section we introduce the model we used to represent data and modifications.

The Titan IDE, where these algorithms are implemented, supports Testing and Test Control Notation - 3 (TTCN-3)[5] and Abstract Syntax Notation One (ASN.1) files [13].

3.1. **Data representation.**

Our representation has one node for every module (TTCN-3 compilation unit) serving as the local root node of the module. These local root nodes form a list of nodes, representing the complete set of semantic information. Modules have subnodes for each top level definition, and definitions can also have subnodes in a recursive manner. This continues till the leaf nodes, which represent some semantic terminals.

The AST stores location information in attributes. The region of each node is textually enclosing the regions of all its subnodes. This means that for any point in the module, there is a node in the AST, whose textual location contains that point and has the shortest region of such nodes.

To provide performance benefits the semantic analysis can reorder the elements of the AST, while keeping their structural information. Where ordering is not defined by the language, the items are stored in associative containers.

3.2. **The model of modifications.**

Our assumption is that a user can change the text within a file at any location. Indirectly applying the following modifications to the AST:

- Creating and inserting new subtrees into the AST.
- Deleting subtrees of the AST.
- Merge or divide nodes in the AST[6].
- Editing text with no semantic value, does not change the semantic state, but locations might need to be adapted.

[5]TTCN-3 [14] is an imperative programming language with testing related extensions and with syntax and semantics close to imperative languages like C.

[6]Insertion of terminals can separate a semantic node into several new nodes. Deleting the closing and opening sings can merge nodes into a new node.

Changes might also introduce syntactical errors that corrupt parts or the whole AST. It is also possible that previous syntactical errors were corrected with a modification of the AST.

We assumed that most of the time users are editing consecutively, working in logical units. This means that most of the time the same or related nodes are changed in the AST.

3.3. **Re-parse points.** To change the AST in a consistent manner, to handle parsing of the minimal amount of text, and to contain the negative effects of syntax errors, special nodes have to be located in the AST. A re-parse point is a node in the AST, whose textual location can be re-parsed as a consistent entity, based on the information known about the AST and the modification as a precondition. These AST nodes form a subtree in the BNF of the language. When the same text representing these nodes is parsed again, the AST has exactly the same semantic meaning.

4. The measurement environment

In this section we introduce our measurement environment, and our measurements of the full file parsing method.

For performance measurments we used 8 projects (Table 1) of different sizes and 2 different execution modes:

(1) Client mode simulates the program starting up in an unoptimized environment. Java VM[7] is started in "client" mode. We measured the first executions of the algorithms.
(2) Server mode simulates an optimized environment. Java VM is started with the "-server"[15] flag[8]. The algorithms were run 5 – 10 times using exactly the same operations before measuring.

To see if the projects and execution modes will give us measurable differences we measured the first syntax checking, which has to read all text in the project. We measured (Figure 1) a clear correlation between the execution time and the amount of text to be processed.

[7]Virtual Machine

[8]uses the most aggressive performance optimizations

Table 1. Projects analyzed

project index	1	2	3	4	5	6	7	8
number of modules	4	39	65	68	118	204	567	828
thousand lines of code	28	19	52	66	66	436	1.174	826

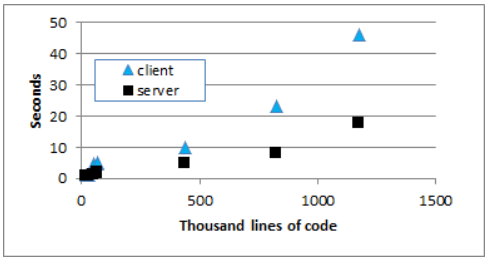


FIGURE 1. Full syntax checking performance by lines

5. The incremental parsing algorithm

In this section we present the methods of the incremental processing algorithm and provide performance measurements.

5.1. **Collecting changes.** In Procedure 1 the processing of syntactic changes and semantic checking is separated. The consecutive changes are merged (line 2) first to reduce the number of longer running checks. The elements of the merged list are processed by invoking a syntax check on them. When all changes are syntactically processed, a single semantic check (line 5) is executed.

Our tool collects changes and does calculations as a background thread to not burden the user interface with heavy processes. Our prototype tool also waits 1 second after the user has finished working on the text to start the background processing. This was seen to be efficient in practice. While the user enters or deletes a consecutive list of characters, the background processing does not start.

5.2. **Refreshing the AST.** The generic form of finding the affected nodes is a recursive algorithm (Procedure 2), invoked with the node to be processed, and the change that happened to the file.

The algorithm iterates on the child nodes of its actual parameter node. The locations of children following the change are updated (line 6). When a change is located inside the child, the algorithm is invoked on it recursively (line 8). If that fails to contain the change, the algorithm reparses the child node itself (line 11). Should that fail, or the change be outside the child nodes, the algorithm backtracks to the previous level of recursion by returning failure.

5.3. **Processing an affected node.** We created specialized versions of the generic algorithm for processing certain types of nodes efficiently. To see how these algorithms interact, see the example in appendix B.

5.3.1. *Processing the file/module root.* The module root[9] has no parent node to regress to, if it could not handle the change the whole file has to be re-parsed (Procedure 3).

5.3.2. *Processing semantic lists.* Semantic lists[10] are elements of the same root type and follow each other in the text. In these structures ordering is not assumed, any number of elements can appear or disappear after a change, and the semantic analyzer can reorder these elements.

The algorithm (Procedure 4) uses two variables, left and right (boundary), to store the locations of safety boundaries. The invariant of this region is that nodes outside of it are not affected by the change and the procedure minimizes this region. It also stores whether the change was already enveloped in a child node: the variable enveloped is initialized to false.

This algorithm differs from the generic in that, first the smallest interval to be parsed is measured by checking the nodes against the changed region (lines 5 – 13). Nodes falling in this region are removed (lines 14 – 18) and the region is re-parsed (line 18) integrating any new node found. If there were no errors, the locations of the nodes following the change location are updated.

5.3.3. *Processing non-list semantic structures with elements.* In semantic structures containing semantically different sub-nodes[11], the ordering and existence of its sub-nodes is fixed[12] and they can not appear or disappear without causing syntactic errors[13]. The children of such nodes are processed in the order of their appearance in the textual representation of the node (Procedure 2).

5.3.4. *Processing semantic structures with no elements.* In terminating nodes, without semantic branches either the whole node has to be re-parsed or it can not be re-parsed, in those cases negative result has to be reported.

---

[9]In our implementation this is the TTCN3Module class, which represents the whole of a TTCN-3 module.

[10]In our implemetation the Definitions class represents the list of all definitions on the module level (types, templates, constants, module parameters, functions, altsteps, testcases). The class StatementBlock represents the consecutive list of statements inside functions, altsteps, testcases, statement blocks and the control part.

[11]In our implementation any semantic node that has children, but does not fit in the first 2 categories. For example: the Def_Function class representing a function definition.

[12]Some of the sub-nodes are required to create a syntactically and semantically correct node, while some can be optional. As an example a function might have a name, formal parameter list, return clause, body (a statement block).

[13]In a function definition if any of the name, formalparameter list (as the whole entity), the *runs on* and *system* clauses, or the statement block(as the whole entity) are damaged, the whole function definition is incomplete.

6. Effect on the semantic checking

The incremental parsing algorithms allows improvements on the performance of the semantic checking as well. Decreasing the amount of changes on AST makes caching of previously calculated results possible. The introduction of version handling for the semantic nodes can provide both safety and performance gain. This can be done by assigning a time-stamp, holding the time of the last semantic analysis to semantic nodes. After the first parsing, all nodes are time-stamp uninitialized, and nodes are checked semantically. Following this check, the nodes can have the time-stamp of the actual semantic check cycle.

A node that was already semantically checked needs to be re-checked only when the semantic properties of the node, it's contents or a referenced node has been changed. As the dependency hierarchy of the modules is known from the previous semantic evaluation and does not change, it can be used to find which modules reference changed modules directly or indirectly.

7. Performance Analysis

7.1. **Searching for the correct node.** Our AST is a tree, where non-leaf nodes can have different number of child nodes. The root of this tree is the semantic node representing the module. The depth of this tree can be estimated with $O(\log_M n)$, where M denotes the branching factor (the children an internal node has in average) and n denotes the number of nodes. This AST has a high branching factor, making the tree's height very small. In modern programming methodology an embedding of more than 7 levels is usually considered a serious design fault, in practice the depth of the tree is usually less than 10.

When the algorithm explores the AST from the root, to the location that closest approximates the damaged region, it is traversing this B-tree, which can be estimated to take $O(\log_M n)$ steps. In the best case, this is the final location, and it is able to process the changes there. This means the processing of $\frac{S}{M^k}$ characters, where S denotes the size of the whole file and k denotes the number of levels descended.

Sometimes it is not able to handle the change on the lowest level, hence the amount of work done by the tool can be estimated with

$$\sum_{i=\log_M n}^{f} \frac{S}{M^i}$$

where f denotes the level where the change could be handled.

In the worst case, when the whole module becomes corrupted by the changes, this means that slightly more work is done as a normal parse does. In this case it had to parse not only the whole file but also had to parse smaller parts of it in order to decide that they are not able to contain the damage done. In the best case only the lowest level of the tree has to be parsed. As the amount of characters to re-parse decreases significantly on each level of the tree, the effective work becomes very small. In the average case, the algorithm finds the node that may contain the damage somewhere in between the corner cases. In some cases the whole file does not need to be re-parsed, which means that most probably it doesn't have to parse more than S/M characters. In such a case the execution time of the program will decrease by some power of M.

Since users usually transform syntactically correct text into an other syntactically correct text by very small changes, a valid assumption could be that the average execution time is close to the one estimated for the best case.

7.2. **Memory usage.** The described algorithms use only local variables needed to keep information like numbers for the left and right boundary, data that is already available in the AST, or can be calculated.

When the re-parsing of the damaged region is done, a part of the AST is rebuilt. Once this new AST part is inserted into the semantic database, the old version is removed.

7.3. **Implementation.** The implementation can be done iteratively. Traversing the tree only to the definition level in the AST already provides a speedup. In this case, when the algorithm finds that the damaged area is completely enveloped by a definition, it re-parses the whole definition. Limiting the amount of text to be processed from the whole file to a single definition decreases the original execution time to $O(s/m)$.

Once the algorithm reaches the level of statements and references, the amount of work has already been decreased to a minimal level. In practice, when general programming design style guides are followed, this should be no more than a single line of text. Decreasing the amount of work to below one line of characters, might not provide any significant benefits[14].

Implementation is not limited to procedural methods, our implementation in *Eclipse Titan* is done using Object-Orientet constructs.

[14]For example the list of formal parameters in the FormalParameterList class could also be treated like a list, but as there are usually only a few formal parameters in a list we choose to not make them incremental yet.

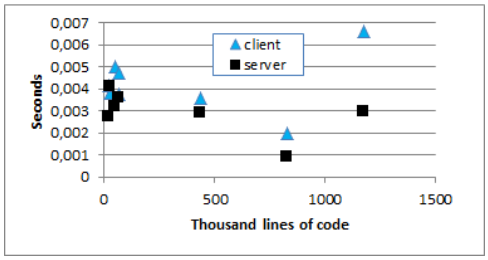


FIGURE 2. Incremental syntax checking performance by lines

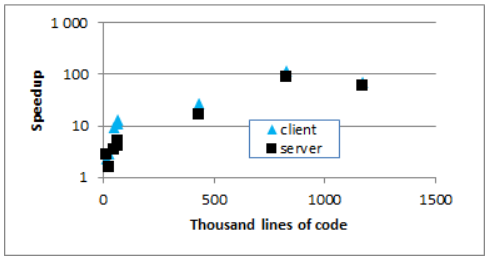


FIGURE 3. Incremental syntax check speedup

7.4. **Measurements.** To trigger the incremental syntax checking, we entered a single space in a randomly choosen module, without changing the semantics. In this setup: (1) the location to be parsed is determined, (2) the incremental parser is invoked, (3) some text is processed and the location informations are adjusted. The amount of characters to be parsed is minimal, minimizing the text processing overhead of the system.

In all cases the execution time of the incremental syntax analysis stays below $7 \cdot 10^{-3}$ seconds in client, and $3 \cdot 10^{-3}$ seconds in server mode (figure 2). In some cases the execution time goes as low as $9,23 \cdot 10^{-4}$ seconds in server mode.

Figure 3 displays the speedup measured showing how much this method improves the performance.

8. CONCLUSION

In this article we presented an algorithm that makes incremental parsing of TTCN-3 files possible, reparsing needs only as much text as necessary to analyze the change. We also demonstrated how this algorithm can be integrated into an existing system. We showed an approach to enhance an existing semantic checking system in order to take better use of incremental parsing.

The measurements shows that the algorithm yields reduced parsing times. With the original method, small modifications triggered the analysis of the whole file, lasting up to seconds. With the proposed algorithm the execution of exactly the same test was hardly noticeable for human users.

9. Further work

It is desirable to continue this work, investigating the efficient processing of the semantic changes. The algorithms described only change states of semantic entities, whose textual representation changed. This should enable efficiently processing of the semantic changes.

It would also be desirable to extend this analysis with supporting other languages. We believe that the algorithms described can be used for efficient processing of other file formats. This still needs to be checked in practice.

10. Acknowledgements

The authors would like to thank the Test Competence Center of Ericsson Hungary for supporting this research and open sourcing the Titan project. The Titan project contains the implementations of the algorithms and is accesible as *Eclipse Titan* here: https://projects.eclipse.org/proposals/titan

References

[1] S. P. Abney, *Rapid incremental parsing with repair* in Proceedings of the 6th New OED Conference: Electronic Text Research, University of Waterloo, Waterloo, Ontario, 1990, pp. 19.

[2] M. Collins and B. Roark, *Incremental parsing with the perceptron algorithm* in Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, ser. ACL 04., Association for Computational Linguistics, Stroudsburg, PA, USA, 2004. articleno. 111, http://dx.doi.org/10.3115/1218955.1218970

[3] C. Ghezzi and D. Mandrioli, *Augmenting parsers to support incrementality* Journal of the ACM, vol. 27, no. 3, ACM, New York, NY, USA, Jul. 1980., pp. 564579, http://doi.acm.org/10.1145/322203.322215

[4] L. Huang and K. Sagae, *Dynamic programming for linear-time incremental parsing* in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ser. ACL 10., Association for Computational Linguistics, Stroudsburg, PA, USA, 2010, pp. 10771086. http://dl.acm.org/citation.cfm?id=1858681.1858791

[5] F. Jalili and J. H. Gallier, *Building friendly parsers* in Proceedings of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ser. POPL 82. New York, NY, USA, ACM, 1982, pp. 196206. http://doi.acm.org/10.1145/582153.582175

[6] J.-M. Larchevêque, *Optimal incremental parsing* ACM Transactions on Programming Languages and Systems, vol. 17, no. 1, ACM, New York, NY, USA, pp. 115, Jan. 1995. http://doi.acm.org/10.1145/200994.200996

[7] G. Linden, *Incremental updates in structured documents*, Ph.D. dissertation, (1993), Department of Computer Science, University of Helsinki, https://helda.helsinki.fi/bitstream/handle/10138/21469/abstract.pdf?sequence=2

[8] A. M. Murching, Y. V. Prasad, and Y. N. Srikant, *Incremental recursive descent parsing* Computer Languages, vol. 15, no. 4, Pergamon Press, Inc., Tarrytown, NY, USA Oct. 1990, pp. 193204, http://dx.doi.org/10.1016/0096-0551(90)90020-P

[9] T. Par, K. S. Fisher: *LL(*): the foundation of the ANTLR parser generator* in Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation, ser. PLDI' 11., ACM New York, USA, 2011, pp. 425-436, ISBN: 978-1-4503-0663-8 doi: 10.1145/1993498.1993548

[10] L. Petrone, *Reusing batch parsers as incremental parsers* in Proceedings of the 15th Conference on Foundations of Software Technology and Theoretical Computer Science. London, UK, Springer-Verlag, 1995, pp. 111123. http://dl.acm.org/citation.cfm?id=646833.708027

[11] T. A. Wagner and S. L. Graham, *History-sensitive error recovery* in In preparation. 24/9/1997 17:26 PAGE PROOFS master, 1997.

[12] Li, Warren X., *A Simple and Efficient Incremental LL(1) parsing*, in proceedings of the 22nd Seminar on Current Trends in Theory and Practice of Informatics, SOFSEM '95 (1995), Springer-Verlag, London, pp. 399–404, http://dl.acm.org/citation.cfm?id=647005.712013

[13] ITU, *Information technology Abstract Syntax Notation One (ASN.1): Specification of basic notation*, International Telecommunication Union, 07 2002. http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf

[14] ETSI, *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language*, European Telecommunications Standards Institute, 04 2012. http://www.etsi.org/deliver/etsi_es/201800_201899/20187301/04.04.01_60/es_20187301v040401p.pdf

[15] Oracle Inc. *Java tuning white paper*, (2005), http://www.oracle.com/technetwork/java/tuning-139912.html

Appendix A. Procedures

Procedure 1 Change processing

```
1: procedure BackgroundThread(List of changes)
2:     List merged ← Merge(changes);
3:     for all change element of merged do
4:         Syntax_check(GetModule(edited_file), change);
5:     Semantic_check(getProject(module));
6: end procedure
```

Procedure 2 The simplified algorithm skeleton

```
1: procedure GenericUpdate(Node node, Interval change)
2:     for all child element of node do
3:         if child.end < change.start then
4:             continue;
5:         else if child.start > change.end then
6:             RecursiveUpdateLocation(child);
7:         else if (child.start < change.start) and (child.end > change.end)
   then
8:             result ← GenericUpdate(child, change);
9:             if result = true then
10:                UpdateLocation(child);
11:            else if Reparse(child) = false then
12:                return false;
13:        else
14:            return false;
15:    return true;
16: end procedure
```

Procedure 3 The algorithm for module level

```
1: procedure Update(Module module, Interval change)
2:     if (module.start < change.start) and (module.end > change.end) then
3:         result ← Update(module, change);
4:         if result = true then
5:             UpdateLocation(module);
6:             return ;
7:     ParseFile();
8: end procedure
```

Procedure 4 The algorithm for semantic lists

```
1: procedure UPDATE(ListNode node, Interval change)
2:     Number left ← node.start;
3:     Number right ← node.end - 1;
4:     Boolean enveloped ← false;
5:     for all child element of node do
6:         if child.start > change.start then
7:             rightBound ← Min(child.start-1, right);
8:         else if child.end < change.start then
9:             leftBound ← Max(child.end, left);
10:        else if (child.start < change.start) and (child.end > change.end)
   then
11:            result ← Update(child, change);
12:            if result = true then
13:                enveloped ← true;
14:    if ¬ enveloped then
15:        for all child element of node do
16:            if (child.start > left) and (child.end < right) then
17:                Remove(child);
18:        enveloped ← Reparse(node, left, right);
19:    if ¬ enveloped then
20:        return false;
21:    else
22:        for all child element of node do
23:            if (child.end > right) then
24:                RecursiveUpdateLocation(child);
25:        return true;
26: end procedure
```

```
1  module Example {
2    ...
3    function demonstration (in integer p_value) runs on
         demo_component_CT
4    {
5      ...
6      for (var integer i:= 0; i <= 10; i := i+1) {
7        ...
8        j := j + i;
9        ...
10     }
11     ...
12   }
13   ...
14 }
```

Listing 1. Example TTCN-3 code

Appendix B. Example operation

The example code (Listing 1) contains one function, with one loop and one statement shown, among any number of other definitions and statements.

If the addition (line 8) is changed to a substraction, the algorithm is called with the module node (line 1) and the location of the changed chaarcter. The module level (Procedure 3) forwards processing to the list of definitions, which determines (Procedure 4) the definition to be checked (line 3). The function definition checks (Procedure 2) it's parts and determines, that the statement-block might be able to handle the change. The statementblock searches the list of statements (Procedure 4) and finds that the loop statement (line 6) might handle the change. The for loop checks it's parts (Procedure 2) and finds that it's statementblock needs to be searched. In the statementblock of the for loop, the statement (line 8) containing the modification is found (Procedure 4).

At this point the tool needs to parse only a single line. Implementing the algorithms to reduce this region further can be a business decision, based on how much the next step in the implementation costs, and how big of an improvement it would bring.

Ericsson Telecommunications Hungary, H-1117 Budapest, Irinyi J. u. 4-20, Hungary

E-mail address: Kristof.Szabados@ericsson.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

TOWARDS SAFER PROGRAMMING LANGUAGE CONSTRUCTS

ÁRON BARÁTH AND ZOLTÁN PORKOLÁB

Abstract. Most of the current programming languages inherit their syntax and semantics from technology of the 20th century. Due to the backward compatibility, these properties are still unchanged, however newer technologies require different language constructs and different semantics. Instead of redefining the programming language, the developers enhance the language with new library functions, or they add some – occasionally ambiguous – elements to the syntax. Some languages provide very loose syntax, which is harmful, because it leads to critical errors. In other case the interleaving "normal" code and exception handling code can obfuscate the developer itself and the subsequent developers.

This paper highlights several aspects of language elements such as basic and potentially unsafe elements of the syntax, control flow constructs, elements used in const-correctness, type-system, elements of multiparadigm programming – generative and functional –, capabilities of embedding a DSL, parallelism support, and taking account of branch prediction. These aspects determine the usablity, safety and learnability of a language. This paper also gives recommendation for a new and safe experimental programming language.

1. Introduction

Current mainstream object-oriented languages contain several problematic constructs which potentially lead to critical errors. Most of the errors came from the loose syntax or not proper semantics. In our work and research we saw a lot of harmful codes, and we intended to give recommendations to extend the coding style or to design a new programming language in order to avoid malicious constructs.

Received by the editors: May 1, 2014.

2010 *Mathematics Subject Classification.* 68N15.

1998 *CR Categories and Descriptors.* D.3.3 [**Software**]: Programming Languages – *Language Constructs and Features*.

Key words and phrases. programming languages, compilers, syntactical vulnerabilities, semantical vulnerabilities, compiler techniques, safe language.

The programming languages have been created in order to reduce assembly errors and to increase the abstraction level. The early languages (e.g. COBOL, LISP, ALGOL, FORTRAN [18, 11]) introduced higher abstraction level, but in this early age, languages were imperfect – they contain many serious issues in syntax and semantics. After the fundamental paradigms invented, the structure of the programming languages started to lose their major gaps in syntax and semantics. And the C programming langauge has beed released [9] in this era.

Nowadays, most of the programming langauges contain syntactic and semantic legacy from the early programming langauges, however, the basic concept is obsolete. The modern processors and computers are designed to execute multiple tasks at the same time, but the early languages did not support that. Because of the huge existing code bases, the developers of the programming languages – for example the C++ – keep the language to be backward compatible. This decision can be debated, and the backward compatibility precludes important security and safety changes in the language.

Our research is based on the alpha version of our experimental programming language, which is currently contains intention to be a syntactically and semantically safe language. The research aims to improve our language, furthermore identify and highlight the vulnerabilities of the current programming languages. This paper includes but not limited to the presented finding.

This paper is organized as follows: In Section 2 we present critical syntactical errors in current mainstream programming languages, we give recommendations about const-correntness and we analyze the loop constructs with examples. In Section 3 we describe main semantical features, we highlight vulnerabilities of the type-system (like infinite loop caused by improper comparisons or unwanted implicit casts) and exceptions (injecting the rarely used exception handler codes into the normal control-thread can mislead the developers and makes the code less understandable), furthermore, we discuss the necessity of the multiparadigm languages (generative and functional language elements, implementing uncommong data structures). Section 4 is about the extended features for support present technologies, e.g. embedding a domain-specific language or writing multithread programs – while staying only at the language features. In Section 5 we overview code generation, like the usage of built-in branch prediction in the programming languages and in the modern CPUs. In Section 6 we shortly brief our experimental programming language. Finally, in Section 7 we describe our development plans. Our paper concludes in Section 8.

2. Syntactical elements

The syntax of a programming language is its face, and it can be judged by the programmers. Because the programmers are the *users* of the programming languages, it is important to keep the syntax as clear, pure and intelligible as possible. Adding loose elements to the syntax can be very harmful, because the reason of some critical errors can be attributable to the loose syntax.

In the following we discuss some syntactical vulnerabilities and improvements.

2.1. **Permissive syntax.** Recently, articles appeared on the Internet about a mistake in a source code belongs to Apple Inc. [23] The reason of the error is a mistakenly duplicated line containing only the `goto fail;` statement – that is why this error called as *goto fail*. This error resulted a serious security leak.

Two problems can be identified in the source code in Figure 1. First, in the *then* branch of the *if* statement we can see only a single statement instead of nested inside a block statement, i.e. came from the permissive syntax of the C programming language. Second, a proper coding style should avoid the use of unconditional jump statements.

```
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
  goto fail;
  goto fail; // duplicated line here
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
```

Figure 1. The affected lines of the *goto fail* error.

Using block statements in every branches of the *if*, *for* and *while* statements is not without precedent – many coding styles require the block statement for reason. Furthermore, adding empty lines makes the code more legible. Although, the block delimiters and empty lines stretch the code, they provide more benefits, e.g. making it more readible.

Applying the recommendations, there is an improved version of the code as can be seen in Figure 2. At this point, the duplicated `goto fail;` statement does not affect the code at all. Even the duplicated line will be ignored by the compiler, so there is no overhead at runtime. Furthermore, the duplicated line is a dead code, so the compiler can detect it, and the programmer can get diagnostics message.

Note that, this feature exists in other languages influenced by the C programming language: C++ [20], C# [1], and Java [4]. In C++, the *goto* statement is also available. However, C# and Java do not have *goto* statement

```
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
{
  goto fail;
  goto fail; // duplicated line here
}

if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
```

FIGURE 2. Transformed version of the error handling.

(as its classical meaning), but the block statement is not required – this can lead to similar error like *goto fail*. Using the `break;` statement is attainable a similer problem.

Because the loose part of the syntax caused the *goto fail* error, we recommend mandatory block statements in the coding style. Even better if the programming language does not allow standalone statement after *if*, *for* and *while*. The syntax of our experimental language requires block statement after the concerned control statements. This is a nice way to avoid errors which are undetectable by the compiler.

2.2. **Constant variables.** In declaration of variables or parameters the programmer should take into account of constness. In C++, the const-correctness deals with which variables or objects are mutable and which are immutable [6]. This is a compile-time construct, so it has to be noted in the source code. The most common way to express the constness is to tag the type with a modifier. In C++, is frequent to pass function arguments as *const* or as *const-reference*.

C++ uses the `const` qualifier to make a variable or an object *read-only*. C# and Java provides a similar mechanism to express constness. Using the `final` qualifier in Java makes the variable unassignable, thus the variable must be initialized in the declaration. Note that, the content of a `final` variable is still modifiable. In C#, the `readonly` keyword has the same effect as the `final` in Java. Furthermore, C# introduced the `const` qualifier, which has an effect similar to the `#define` in C++. However, this feature has some shortages, for example parameters can not be marked as `const`.

As we have seen, in current mainstream languages the default behaviour is the mutable. Notable counter-example the C++11 lambda functions [8], where the default is the `const` mode. The lambda is a syntactical sugar, and compiled as embedded structs. By default, the generated functor is a constant member function. This is also an example, where a new syntax element do not have to be backward compatible with earier elements. Since it is a new language construction in C++, it uses a safer approach.

The Version 2 of the D programming language supports two different aspects of constness. A variable can be mutable and `immutable`, the view of a mutable variable can be `const`, and the view of an immutable variable can be `const` as can be seen in Figure 3 [21].

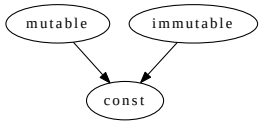


FIGURE 3. constness relationships in D programming language.

Unfortunately keeping the const-correctness of the code is not simple. In C++, almost all code can be compiled without using any `const` qualifier. And based on the fact that programmers do not want to write more than necessary, the `const` will not be placed everywhere where needed. An other problem is to teach and to account const-correctness.

Let approach this problem from a totally different aspect: do not require `const` qualifier for immutable variables, but require the `mutable` qualifier for variables and arguments which can be modified. Using this technique, the compiler will enfore to keep the const-correctness of the code. At this point, the programmer can forget only the `mutable` qualifier, but the compiler will calling to account the missing qualifier. Therefore, the code will be more safer thanks to the inverted psychology – programmers can not neglect mandatory keywords.

Parameters	236
By value	87
Constant	105
Mutable	44

FIGURE 4. Function parameter analysis of TinyXML.

Might arise the question that more or less keywords must be used in the code. An analysis about TinyXML 4 can be seen in Figure 4 – the rate of the constant function arguments is about a half to the total number of arguments. However, the analysis showed, the parameters passed by value is relevant. This implies that, the number of mutable arguments (*total number of arguments - constants - by value*) is less than half of constant arguments. Based on this analysis, using our suggestions half of qualifiers are required, and the compiler will check the const-correctness (via requiring the `mutable` qualifier instead of `const`).

2.3. **Control flow.** Sometimes the programmer needs to understand someone else's code. It is a great help if the artifacts belong together are collected in one group. The most common issue in this topic is the `while` and the `for` loops. Almost every programming language supports at least two different loop construction, and the `while` and the `for` loops are sufficiently widespread.

The key problem with a common `while` loop is the loop-variable. Three activities can be identified: the initializing-, the role in the loop-condition-, and the stepping of the loop-variable. These pieces can be scattered into the statements of the `while` loop. Obviously, the code is obfuscated, and this is only the programmer's fault.

This implies the problem: how to force the programmer to use the proper loop construction. Take an example from a C++ code: given a variable of `std::vector<int>` [12, 2, 15], print its the content to the standard output. Strive to use only the language features. The first construction in Figure 5 can easily mislead the programmer who would like to understand the code.

```
int idx = 0;
while(idx<vec.size())
{
  std::cout << vec[idx] << std::endl;
  ++idx;
}
```

FIGURE 5. A poor implementation of iterating an array.

Therefore, gather the related pieces together: use the `for` statement and place the three parts next to each other. The new construction in Figure 6 is completely understandable, but not the best solution, because the code uses the indexing operator, thus the code is less portable.

```
for(unsigned int idx=0;idx<vec.size();++idx)
{
  std::cout << vec[idx] << std::endl;
}
```

FIGURE 6. An other implementation of iterating an array.

The modified code satisfy the portability requirement, because the it can be generalized easily. Unfortunately, the code became larger and more complex as can be seen in Figure 7. This piece of code uses the the iterators of `std::vector<int>`. This is not so comfortable, and must be replaced with a better construction.

```
for(std::vector<int>::const_iterator
  it=vec.begin(),end=vec.end(); it!=end; ++it)
{
  std::cout << *it << std::endl;
}
```

FIGURE 7. Using the iterators of `std::vector<int>`.

The final solution came from the idea that why the programmer should write a simple iteration again and again every time, while the construction is quite the same. Also, writing more code means more possibility to make a mistake. The related code can be generated automatically by the compiler. The C++11 standard [20] introduced the *foreach* mechanism, and its usage and the final version of the code can be found in Figure 8. The *foreach* mechanism is available in other languages such as Java, C#, go, python, etc.

```
for(int x : vec)
{
  std::cout << x << std::endl;
}
```

FIGURE 8. A safe implementation of iterating an array.

This is quite good and safe solution, because the programmer wrote what container has to be iterated, and not how the container has to be iterated. We recommend that to use the *foreach* mechanism where possible, and avoid the fragile and misunderstandable constructions.

2.4. **Summary.** As can be seen above, restricting the syntax can improve the code. It is important to declare additional rules in the coding conventions, for example "use braces in all control statements". In case of designing a new language, it is a good start to require explicit compound statements to avoid malicious contructs. Also, the need of the `goto` statement must be considered – high-level languages rarely support it.

Based on the presumption, that programmers are lazy, the usage of the reversed philosophy of the constness improves the quality as can be seen above. Even if the changed syntax may result longer code, the quality improvements are more beneficial.

3. Semantical elements

The following section describes the used type-systems in programming languages. The aim is to uncover the vulnerabilities of the constructs, and give a safer alternative implementation. We used these alternatives in our experimental programming language.

3.1. **Type-system.** There are two different approach of the type-systems. The first is the completely dynamic type-system, where the variables do not have types in the source code. This type-system is commonly used in script languages and in some functional languages, for example in Erlang [17]. The dynamic type-system is a new opportunity for freedom, but understanding the code (i.e. static analysis) is much harder. The second is the static type-system, which is widespread in imperative- and object-oriented programming languages. The static type-system is used to express the intended type for every variable. It helps to understand the code. This also implies an additional check by the compiler. In the following we assume that, the programming language uses static type-system.

Even the static type-system is fairly safe, problems are hidden inside. For example, the implicit cast mechanism is sometimes useful, but it can result very harmful situations, such as infinite loops, and hardly followable or unpredictable code behaviour.

In the C++ language we can define constructors with one parameter. Without the `explicit` keywords these contructors can be called implicitly as conversion. Thus, the compiler can create, for example, complex object from a single integer. Moreover, C++ classes can also define conversion operators, and classes can be converted to primitive types without explicit cast. These features can mislead the programmer who wants to understand the source.

```
// [1]
std::vector<int> vec;
for(int i=0;i<vec.size();++i) { /* ... */ }
// [2]
unsigned int x = 0xffffffffu;
if(-1 == x) { /* ... */ }
```

Figure 9. Comparison of signed and unsigned integers (C++).

In C++, semantically wrong comparisons can be written, but the compiler indicates only a *warning* – however, there are major semantical errors. The code in Figure 9 contains two semantical errors. The first piece is a common error: comparing a signed and an unsigned integer can outcome bad result.

The compiler will show a warning for this part. The second piece is worse than the first one, because checking the equality of an unsigned an signed value is meaningless. The problem is that, the representation of the largest unsigned value (`0xffffffffu`) and the *minus one* are the same. The compiler accepts the code, the condition of the *if* statement is true, but no warnings and no errors are generated.

The origin of these problems is that the signed and unsigned intergers have different domain, although the half of the domains overlap. There are two solutions for this problem. For example, the Java programming language does not introduced unsigned integers. Still the implicit casts can ruin a Java program as well: the code can be seen in Figure 10 is an implicit infinite loop. Another solutions is not to allow comparisons between different types at all – our experimental language uses this solution.

```
// Valid code in Java, C++, and C#
for(char ch='\0';ch<70000;++ch) { /* ... */ }
```

FIGURE 10. Infinite loop caused by implicit cast.

3.2. **Multiparadigm support.** The languages which support multiparadigm programming are more flexible, because the programmer can use the proper paradigm for the specific task. The most conspicuous is the C++11 standard, because of support for generic programming (especially template metaprogramming) [7], object-oriented programming, and functional programming. In C++, the functional programming is observable in the template metaprogramming, and in the lambda expressions.

The highligthed feature is the lambda expressions: introduced as a useful feature, but it can easily go wrong. Basically, the lamba are used to replace unnecessary boiler-plate codes with a compact syntax. Due to these lamba are very sematic, the compiler can generate the affected embedded structs – the result is the same, but with a safer syntax. However, when the programmer writes larger lamba, the understandability of the code is heavily decreasing.

Furthermore, the *union* contruction of the C++ can be connected with the algebraic data types from functional languages. Although, the *union* can be implemented with abstract class and derived classes, it needs a massive amount of code. The task is to store different values, and to handle them as uniform. Staying as close to the language features as possible, we do not take account the `Any` from the *boost* library. There is a huge problem with the *union* construction: it is possible to "cast" a character to a pointer at any time without any compiler feedback. However, the algebraic data types are much safer than the *union*, and it is a closed type. The common *tuple*

type (for example in Erlang) is an open type, the programmer can create a completely new variant of a tuple without modifying the type declaration – but not with the algebraic data types.

Consequently, the algebraic data types could be a powerful feature in C++. This gave the inspiration to us, to implement the algebraic data types in our experimental language.

The generics in Java programming language is a syntactical sugar introduced to help the programmer to write type-safe collections and type-safe classes [5, 4]. Most of the Java programs before JDK 1.5 contained massive amount of basically unnecessary casting, and a `ClassCastException` could be thrown. However, after the program passed the static analisys phase, the compiler omit this type-info from the class file – this itechnique is called *type-erasure*.

3.3. **Exceptions.** As mentioned above, understandibility and clear-looking code is important. Another critical topic is the exception handling. The exception is raised or thrown when an exceptional event happened – for example: program ran out of memory, or an index is out of range. The program must be prepared to handle the exceptions, but these parts of the code may not executed at all. However, some languages, like Java, C#, and C++, allow – and requires – interleaving of "normal" and exception handler code. Thus, the train of thought have breaked many times, and the programmer can follow the code hardly. This effect appears most frequently in Java and C#.

The idea of the solution came from the Eiffel programming language [14]. The Eiffel supports exception handler code only at the end of the function. In the exception handler code the programmer can write the `retry` instruction to restart the execution of the function.

If the exception handler code is placed at the end of the function, the programmer will enforced to write shorter functions – which is good, because large functions are hardly understandable. Although, in Java, is very unusual and uncomfortable to write the exception handler codes only at the end of the function. However, this technique is a nice way to keep the code organized. Our experimental language uses this technique for placing exception handlers in order to keep the understandability of the source code.

3.4. **Summary.** Based on the presumption, that programmers tend to write the possible minimal code, the loose static type-system can be very harmful. The implicit casts and mixed exception handlers can decrease performace and the understandability. Using a strict static type-system and a strict exception handler syntax results a clear code and improves the code quality.

4. Extended features

Nowadays, it is a requirement to a programming language to support special features, for example embedding a domain-specific language, or writing parallel programs. It is important to use only the features provided by the language, and work without any external tool (script for preprocessing, third party tool) or library in order to keep the portability of the code [16].

Moreover, it is certain optimization to use the features provided by the compiler to implement a domain-specific language or a parallel program, because the compiler is used to know the best solutions for the specific platform.

There are many ways to embed a domain-specific language (DSL) into an existent language [19]. We used the type-system and the operator overloading mechanism to create DSLs. The reason was obvious: the compiler can check all of the expressions, and discover any problematic statement – using only the strong static type-system. The usage of the operators is a convenient feature, because the syntax of the created DSLs are similar to the original language.

In a lot of programming languages the support of parallelism is not part of the language. For example, C, C++, and Pascal can handle threads with library functions. An other example is the C# and Java languages, because these provide a `Thread` class, and the programmer can easily write multithreaded algorithms. Not that, the `Thread` class is not considered as a library, because this is the part of the basic runtime library, and no C#, nor Java program can be executed without it.

There are more possible ways to implement the thread model in a virtual machine [10]: all the threads are an operating-system thread, or some of them are mapped to an OS thread, maybe none of them. Our virtual machine uses a dynamic virtual-thread mapping mechanism – run the thread on the first idle OS thread (in the virtual machine's terminology, the OS threads are the "processors"). Due to this approach, creating a new thread inside the virtual machine does not require massive amount of resources.

4.1. **Summary.** Using a multiparadigm or easily extendable language can keep portability of the code, because no external tools are required. Concurrent programming is essential in modern software technology, therefore the language support of multithreading is important. This trend can be observed in the C++ programming language, since the C++11 standard introduced the threads in the standard library.

5. Code generation

The native programs can be accelerated by the features of the CPU, for example the branch prediction or return address prediction. These features are

built-in into a modern processor to speed up the execution. The results can be seen in Figure 11. The related codes aim the branch prediction feature of the compiler and the CPU. The programs compiled without any optimizations (but using built-in branch predition). This is used to enable runtime CPU optimizations.

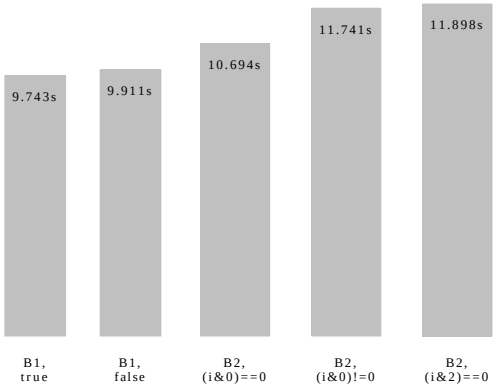


FIGURE 11. Results of the branch prediction test.

```
void sum(bool cond)
{
  int sum1 = 0, sum2 = 0;
  for(unsigned int i=0;i<0xffffffffu;++i)
  {
    if(cond) { ++sum1; }
    else     { ++sum2; }
  }
}
```

FIGURE 12. Code used in B1 test (C++).

Note that, the type and the value of the `sum` variables in codes in Figure 12 and in Figure 13 is not relevant.

However, the codes above after minor changes (upper limit is decreased to `0x7fffffff`) has been used in Java programs too, but the Java compiler and the Java Virtual Machine (JVM) uses massive optimizations. Therefore, there are no difference at runtime with the code in Figure 12. Interesting results came from the code in Figure 13, because when the condition is potentially true (condition is `(i & 0x0)==0`), the Java optimized the whole loop out –

```
void sum()
{
  int sum = 0;
  for(unsigned int i=0;i<0xffffffffu;++i)
  {
    if(<COND>) { ++sum; }
  }
}
```

FIGURE 13. Code used in B2 test (C++).

resulting almost zero time in execution. But, when the condition is not trivial, the execution of program took 3.152 seconds.

It follows that, there are benefits to compile to a virtual machine like JVM, because more runtime optimizations can be applied. Note that, the measurement above is artifical, and may not appear in product code.

5.1. **Summary.** There is several ways of program portability. First, compiling the same code on multiple platforms, and the compiled executable performs the same actions – as the C and C++ languages. Second, compile the source code once, and use the same executable on multiple platforms. This technique requires a virtual machine, or a higher level compiler (JIT). There are arguments in favor of both techniques.

6. WELLTYPE

Our experimental programming language is called *Welltype*[1]. Welltype is a strongly typed imperative programming language, which is based on C and C++ languages, also influenced by Ada and Eiffel languages. The main concept is to minimalize errors caused by typos, type mismatches, unclear control structures, etc. This is ensured via the strict and clear syntax, and the strong static type-system. Some language constructions cause more verbose source code, but the langauge prefers safety over compact code. For example our language handles assignments as statements instead of expressions, like C-style languages to avoid unwanted side effects at assignment.

This programming language is designed to include all recommendation in this paper, and we rendered an example language to show that our findings can be used in practice. The prototype implementation of this programming language is available at `http://baratharon.web.elte.hu/welltype`.

[1]The *Welltype* is an intentionally wrong abbreviation.

6.1. **Permissive syntax.** Our language is invulnerable to the *goto fail* error, since requires braces in every control statement constructions, such as `if`, `while`, `do-while`, and `for`, also there is no `goto` statement in the language. These restrictions are aim the clear and structured programs.

6.2. **Constness.** The Welltype language uses the reversed constness philosophy described earlier. All object parameters are passed as immutable references, but the `mutable` modifier can be placed to make the content of the parameter mutable.

6.3. **Type-system.** In order to keep the type-system clear, our language uses strong static type-system. This implies that, there is no automatic type casting, and implicit object creation. Therefore, all function calls and assignments can be validated by the programmer, since it is based on the static types and the validation is a straightforward algorithm. Every expressions and literals have exactly one type at compile-time, and that type can be converted only explicitly.

6.4. **Multiparadigm support.** We are working on to introduce the algebraic data types and the generics in our programming language.

6.5. **Exceptions.** Our programming language supports an Eiffel-like exception handling: the exception handler can be placed at the end of the function, therefore the exception handler will not interleave with the "normal" logic.

6.6. **Extended features.** The preceding version of the Welltype language was used to implement embedded domain-specific languages (eDSL). We find out, the implemented operator overloading mechanism is adequate for several features, for example implementing DSLs [3].

7. Future work

Based on the findings described in this paper, we will improve the alpha version of our experimental language. The most important intention is the make the syntax and the sematics as safe as possible, and to detect the most of the errors. We will analyse the modified syntax and semantics of our language to find out the attainable vulnerabilities.

In order to ensure about the recommendations we made, we intended to make a personal survey with programmers and non-programmers as well. The survey will consists of two parts: writing a program individually to test the constructions of the language, and adding a new feature into an existent code to test the comprehensibility of the language. By way of comparison, we will use beside our language C++ or Java.

8. Conclusion

In this paper we inspected the safety of different aspects in programming language, for example syntax, semantics, extended features and code generation. We discovered that, the originate of many vulnerabilities and harmful elements is the legacy of the 20th century and sometimes the backward compatibility. We have given recommendations to avoid harmful constructs and to keep the safety and understandability of the code.

The analysis started with the syntactical elements, and we can see that, many problem can be traced back to the loose syntax of the language. This is mainly occur in C programming language and other languages with the similar syntax. In other hand, the proper accessibility notations of the objects (variables or parameters) have added value – in C++, means const-correctness. Due to the permissiveness of the compiler, the semantically missing `const` qualifiers will not be reported. However, the reversed approach can eliminate the problem, and provides more compile-time checks.

The next step was the analysis of the semantical elements. The main aspects was the deficiency of the type-system, e.g. in C++, C# and Java, the programmer can easily write infinite loops without any compiler feedback. The cause of this unfortunately problem is the implicit casts. In our experimental language there are no implicit casts to avoid these kind of errors. We showed that, the exception handlign in mainstream object-oriented languages can break the normal execution thread in the source code, causing harder understandability.

We discussed the opportunity and features of embedding a domain-specific language into an existing host language. In our experimental language we prefer to use the type-system and the operator overloading mechanism instead of external, third party tools. Because our language does not support implicit casts, there are no chance to break the DSL's types unintentionally. An other perspective is about writing multithread program with langauge support only. For example, in C and C++ there is no language support to handle threads.

Finally, we tested the basic built-in branch prediction mechanism in the compiler, and runtime impacts by a modern CPU. We experienced that, the compiler assumes that, the condition of the branch is always true – including *if*, *for*, and *while* statements. It implies that, the programmer should take the branch prediction into account.

References

[1] Albahari, J. and Albahari, B.: C# 4.0 in a Nutshell: The Definitive Reference, O'Reilly Media, 2010

[2] Austern, M. H.: Generic Programming and the STL: Using and Extending the C++ Standard Template Library, Addison-Wesley, 1998

[3] Baráth, Á., Porkoláb, Z.: Domain-Specific Languages with Custom Operators, To appear in proceedings of *The 9th International Conference on Applied Informatics*, 2014
[4] Bloch, J.: Effective Java (2nd Edition). Prentice Hall PTR, NJ, USA, 2008
[5] Bracha, G., Odersky, M., Stoutamire, D., Wadler, P.: Making the Future Safe for the Past: Adding Genericity to the Java Programming Language. In *OOPSLA '98 Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, Pages 183–200, ACM New York, NY, USA, 1998
[6] Cline, M. P., Lomow, G. and Girou, M.: C++ FAQs, Pearson Education, 1998
[7] Czarnecki, K., Eisenecker, U. W.,: Generative Programming: Methods, Tools and Applications, Addison-Wesley, 2000
[8] Järvi, J., Freeman, J.: C++ lambda expressions and closures. Science of Computer Programming 75.9 (2010): 762-772.
[9] Kernighan, B. W., and Ritche, D. M.: The C programming language. Vol. 2. Englewood Cliffs: prentice-Hall, 1988
[10] Manson, J., William P., and Sarita V.: The Java memory model. Vol. 40. No. 1. ACM, 2005
[11] Metcalf, M., Ker Reid, J., and Cohen, M.: Fortran 95/2003 Explained. Vol. 416. Oxford: Oxford University Press, 2004
[12] Meyers, S., Effective C++, Third Edition, Addison-Wesley, 2005
[13] Meyers, S. Effective STL - 50 Specific Ways to Improve Your Use of the Standard Template Library, Addison-Wesley, 2001
[14] Meyer, B.: Object-Oriented Software Construction, 2nd Edition, Prentice Hall, 1997
[15] Pataki, N., Szűgyi, Z., Dévai, G.: *C++ Standard Template Library in a Safer Way* , In Proc. of Workshop on Generative Technologies 2010 (WGT 2010), pp. 46–55.
[16] Porkolab, Z., Sinkovics, Á.: Domain-specific Language Integration with Compile-time Parser Generator Library. In *Proceedings of the Ninth International Conference on Generative Programming and Component Engineering*, Pages 137–146, ACM New York, NY, USA, 2010
[17] Laurent, S- St.: Introducing Erlang, O'Reilly Media, 2013
[18] Schach, S. R.: Object-oriented and classical software engineering. Vol. 6. New York: McGraw-Hill, 2002
[19] Sinkovics, Á, Porkoláb, Z.: Domain-Specific Language Integration with C++ Template Metaprogramming. Formal and Practical Aspects of Domain-Specific Languages: Recent Developments. IGI Global, 2013. 32-55. Web. 30 Apr. 2014. doi:10.4018/978-1-4666-2092-6.ch002
[20] Stroustrup, B. The C++ Programming Language, 4th Edition, Addison-Wesley, 2013
[21] const(FAQ) - D Programming Language. http://dlang.org/const-faq.html#const
[22] TinyXML, http://www.grinninglizard.com/tinyxml2
[23] Vulnerability Summary for CVE-2014-1266. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-1266

Department of Programming Languages and Compilers, Faculty of Informatics, Eötvös Loránd University
E-mail address: {baratharon|gsd}@caesar.elte.hu

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

USING CONCERN SPACES TO MEASURE REQUIREMENTS SIMILARITIES

CĂLIN EUGEN NICOLAE GAL-CHIŞ

Abstract. The software artefacts are crucial during the development cycle of a software product and tracing them is important to the development process. The model used, the requirements document, and the code, are artefacts that can be updated or reused in different projects. Different types of notations are used to add traceability to artefacts, providing versatility in searching, indexing, updating, or retrieving them.

MultiCoS is an approach based on separation of concerns (SoC) in multiple spaces. The concern spaces are defined by grouping concerns by common interest. The relationships between concerns and different types of entities empowers the concern to provide a degree of meaning to an entity. Defining and using concerns to properly describe software artefacts can add semantic to documents such as the specifications document, requirements document, project documents, and to other artefacts such as code files or modules. Given this, the concerns and their relationships can provide traceability to higher level entity spaces, such as the application model, the views, and the design documents of a software application.

The MultiCoS metamodel is presented here in a case study, reusing web applications artefacts in the software development cycle. In addition to other tracing methodologies, MultiCoS can add semantic value to artefacts and can strengthen the relationships to concerns or between artefacts by taking into account similarity coefficients.

In contrast to other methodologies, MultiCoS supports complex tracing systems by creating multiple relationships of different degrees between entities, based on the value of a concern, a subunitary value that measures the impact of a concern to an entity.

1. Introduction

Aspect Oriented Software Development (AOSD) [10] methodologies and Separation of Concerns (SoC) [15] approaches in software development are

Received by the editors: May 1, 2014.

2010 *Mathematics Subject Classification.* 68N30.

1998 *CR Categories and Descriptors.* D.2.1 [**Software Engineering**]: Requirements/Specifications – *Methodologies*; D.2.13 [**Software Engineering**]: Reusable Software – *Domain engineering*.

Key words and phrases. Concern Spaces, Requirements Engineering, similarities metric.

supporting and controlling the development process, in a way that provides a better understanding of the relations between concerned entities. Such a need is mentioned early in software development [3]. Leaded by the relations between concerned entities, the programmers can understand the impact of their work in the final product, and the stakeholders can trace easier how product specifications and requirements were implemented into the software product. MultiCoS separation of concerns approach offers a solution on coping with complexity and with cross-cutting concerns, also being able to provide support for code reusability and reverse engineering.

In terms of requirements specification, in the initial stages of the software development, every approaches is providing and using their own proprietary methods of describing the relations and dependencies between requirements. In addition to providing a method for managing requirements that is suitable to be used with any approach, MultiCoS can add a list of attributes to the requirements. These attributes are providing meaning to the requirements in the context of the software development process. Later in the development process, these atributes can also be used to describe artefacts such as the project documents, the code and other entities interacting with the software product. In fact, these attributes are the *concerns*, and these concerns are grouped into *concern spaces* by their matter of interest.

Some concern spaces will address generic attributes, while others will be specific. A set of default Concern Spaces can be used to properly describe the basics related to the software developement process. Other standard libraries of concern spaces are meant to describe different aspects of the software product and of the parties involved in the software development process. Custom, new concerns can be created and used as needed to fully describe the desired attributes, and concern spaces can be set to encapsulate these new concerns.

In this paper we will show that, while there are certain modelling approaches using concerns to describe specific entities and relations in the software development process, the MultiCoS approach uses a general-purpose concern modelling capability to support the software development process in various aspects and also provides a proper environment for code reusability or software reengineering.

In Section 2 will be presented the existing methodologies based on different approaches using separation of concerns. In Section 3 the MultiCoS approach will be presented, with the primitives used here regarding Concern Spaces, a mapping function used to relate different artefacts and entities to Concern Spaces. This paper's main contribution: a metric and a process that will be used over the mappings to determine artefacts similarities, will also be introduced in Section 3. In Section 4, a Study Case will be conducted. In the end, there will be presented the directions for further work and the conclusions.

2. Related Work

The main venues related to this research paper are the methodologies based on SoC and AOSD. They have introduced the core principles of the concerns and have also presented the goal of using them into the development process.

The need of formalizing the concern spaces is observed in [9]. In the same paper, the relation between concern spaces and units is expressed using graphs, and, for this purpose, a tool is provided to visualize the graphs. To use all the concerns in spaces, even when some concerns do not apply to certain *units*, the author is proposing to map the units using a default *null*. Also, the author debates if the relation between concerns and units, should be with no restrictions, should be an injective relation or a surjective relation (for each, the *units space* is the function domain).

SoC is used in terms of organizational concepts, instead of programming concepts in [1]. Using this approach, the authors are focusing on closing the semantic gap between a software system and their operational environment. *Tropos*, the methodology they use in software development, is based on modelling early requirements using concepts. Tropos comes with five concern spaces *classes*: actors, resources, hard goals, soft goals and tasks.

The software concerns are modeled in *Cosmos* [16]. The primitives of this approach are concerns, relationships and predicates. There are two large concern categories in Cosmos: Logical and Physical. The logical ones are used to describe concepts related to a system or artefact (examples, issues, features, properties), while the physical ones are pointing to entities of the system or to software artefacts that can be related to logical concerns. There are five logical concern categories: clasifications, classes, instances, properties and topics used to express concerns like functionality, behaviour, performance, robustness, state, coupling, configurability, usability, size, and cost. The "real world" entities of a system: hardware, software, subsystems, services are part of the Physical Concerns. They are divided in three categories in Cosmos: instances, collections (as collections of instances or collections of subcollections) and attributes (attributes of instances or collections).

The ModelSoC approach [8] applies SoC to all artefacts, referring initially to documents, models and code. They use their own *hyperspace* model for a multi-dimensional SoC in order to trace data reused in other models. The *Reuseware* framework has been provided as a tool to relate the information available during the development process to a concern space and generate different views and a final version of the system. The multiple dimensions are applied here to artefacts, and not to concerns.

In another approach [14], *hyperspaces* are used to create a SoC on multiple levels. The *units* are separated on means of different facets of a concern,

providing support to an efficient remodularization processes. As envisioned in this approach, concerns are flexible, being capable of interactions and overlappings. Still, not all entities can be related to hyperspaces, limiting the approach to just some entities.

Another paradigm [17], emphasizes the role of the software artefacts and supports the modelling and implementation process of the artefacts. The approach separates cross-cutting concerns in multiple dimensions by using operations of composition and decomposition. The approach is extended in [6] and is part in all stages of software development, providing support for flexible and traceable artefacts.

Requirements are modeled by using SoC in [2]. The application model is transformed, being divided into chunks. When the transformation of the model is completed, the SoC is used to model and transform requirements in an iterative, complete manner and link them to the application.

The Multi Dimensional SoC approach [12] is used in dealing with requirements. The traditional approach of representing requirements using viewpoints, use-cases is dropped in favor to a conceptualized approach that treats equally functional and non-functional requirements. The requirements space is viewed from two logical perspectives, the system space and the meta-concern space. The System Space consists of all existing requirements, so when discussing an application, the application requirements are already included in the System Space. The concerns in the concern space are typical sets of concerns that can be found in various systems. The relation used to map concerns from the Concern Space to requirements of the System Space in an injective function.

The ARCaDE (Aspectual Requirements Composition and Decision Support) approach [11] treats *PREView* concerns as aspects and separates them when dealing with the requirements. The approach supports concern and candidate aspects identification and specification. Also, candidate aspects are composed with the cross-cutting viewpoints.

The *Aspect-Oriented Software Development with Use Cases* approach [7] handles cross-cutting concerns using Use Cases as overlays on top of classes. The technique proposed manages the cross-cutting concerns individually and composes them to obtain the solution system. The approach is influenced by AspectJ and HyperJ vocabulary.

3. MultiCoS Approach

The MultiCoS concept was introduced in [4] and was extended with the approach and the primitives in [5]. MultiCoS is based on the approach presented in [12]. Compared to other methodologies based on aspect oriented or

based on SoC, the current approach offers versatility, applying concerns on any type of entities, logical or physical, to humans and technology involved in the project, to artefacts including, but not limited to project documents, requirements and code. In the following, we will present the primitives of the approach and the mappings that can be established using the primitives. Then, the main contribution of this paper will be presented: a metric to calculate the *similarity* of different entities. The *similarity* will be established by *similarity indices*, which are calculated using mappings performed to the entities over some common Concern Spaces.

The benefits of the proposed approach over the similar existing ones is that this approach, as will be proved next, can be used with any concern related approach, providing a general-purpose concern modelling capability, that brings the possibility to compare approches, using the primitives of the MultiCoS. Being a generic approach, MultiCoS can be applied to a wide range of situations, not related only with software development.

Primitives. The primitives used are: concern, concern space, entity, entities space, concern value, multispace.

- The *Concern Space* is a group of concerns with cohesion, reffering to/ describing similar properties of certain type of entities. The Concern Space sets a range of values to their concerns [0, max_value]. The concern space has to include at least one concern (dimension).
- The *Concern* is one of the dimensions in a concern space. A concern has the goal to describe the attributes of an entity by assigning a corresponding value from the range to the relation. The relation expresses how much the entity is concerned of a descriptor of its attributes, quantifying the importance of the concern to the entity.
- The *Entity* is an object that can be described using a mapping to different concerns in one or more concern spaces. Examples of entities are: requirements, artefacts, users, software modules, technologies, client specifications, even other abstract objects like concern spaces.
- The *Entities Space* is a collection of entities with cohesion, such as the requirements space, viewpoints space, developers space. Sometimes is referred as *System Space*.
- The *Concern Value* describes the strength of the relation between a concern and an entity. The value reflects the weight of a concern to the entitiy, compared to the other dimensions of the same concern space.

Mappings For every entity, there is a vector from each Concern Space that is associated with the entity, expressing the relation of the entity to the Concern Space. This association can be expressed as a function with the

entity and the concern space as the arguments, and with the associated vector of concerns as a result. Let e be an entity in the system space E, S a concern space with n dimensions and f a mapping function that describes the relation between the entity e and the dimensions in the concern space S. Given this, we can write that:

$$f(e, S) = v, \tag{1}$$

where $e \in E$, $v \in R^n$, and v is the vector with the position coordinates of the entity e in the concern space S.

The concern space S is described using the following format:

$S = (id, name, description, list\ of\ concern\ dimensions, max\_value)$

The id is a unique identification label, $name$ is the unique name of the concern space, the $list\ of\ concern\ dimensions$ is the list of the n concerns of the space. An entity can be assigned a value for a dimension in the space in the range [0, max_value]. max_value is the maximum value an entity can get for any dimension in the space, and is a positive, non-zero value.

In [5] was demonstrated that the MultiCoS approach can be used to represent the concern-entity relation in other methodologies based on separation of concerns. The unitary notation provided can replace the primitives used in other investigated methodologies, which makes easier to compare the methodologies. Also meaning is added to the process represented and can be related to a wide range of meta systems.

$ModelSoC$ approach [8] is extended from the hyperspace model introduced in [14]. The approach is applied on Model-Driven Software Development (MDSD), where the models of the system are the artefacts presented in different formats (views). During the process, the model is composed using DSMLs (Domain-Specific Modelling Languages) and delivered to five different viewpoints (such as diagrams, documents, code) by 12 different composition systems (controllers). These composition systems are: (usecase, participation, exchange, flow, trigger, factory, class, dataclass, associate, typebind, app, security [8]. The relation between viewpoints and composition systems, can be expressed in MultiCoS terms, using the compositioon systems as concerns and the viewpoints as entities.

S_0 = (0, Composition Systems, Composition Systems of concerns, (usecase, participation, exchange, flow, trigger, factory, class, dataclass, associate, typebind, app, security), 1)

The mapping function f using entities from the ViewPoint System Space as the first argument and the Concern Space S_0 as the second will give us the following *coordinates* for the entities in the S_0 concern space:

$f(OpenOffice, S_0) = (1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$f(UML\ use\ case, S_0) = (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$f(Value\ Flow, S_0) = (1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0)$
$f(UML\ class, S_0) = (0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0)$
$f(Java, S_0) = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$

We can notice that for each entity above, in the mapping vector there is more than one non-zero value, meaning that the concerns with non-zero values impact the entity on a certain degree. Concerns which impact different entities are potentially cross-cutting concerns in relation with the entities.

Metrics We can consider that entities are *similar*, if the f function gives *close* vectors for entities in concern spaces. A metric will be used to measure the similarity of two entities.

Let there be two entities a, b in the entity space E, $a, b \in E$, and a concern space S_1, with n dimensions. In first place we will measure the similarity of the two entites, a and b, regarding the value of the f function for the two entities in the concern space S_1.

For each dimension i in the concern space S_1, we will calculate ds_i, the *dimension similarity coefficient*, based on three values, two values are a_i and b_i, the entities' corresponding values in dimension d_i and the max_value set for the concern space S_1.

$$ds_i = 1 - \frac{|a_i - b_i|}{max\_value} \tag{2}$$

Given that, for two null values, $a_i = b_i = 0$, we get $d_i = 1$, meaning *maximum similarity*, we will exclude such *dimension similarity coefficients* from further calculations, taking into consideration that the lack of a particular property for two entities can not lead to a strong *similarity*. Let us consider to have p_1 *dimension similarity coefficients* for which a_i, b_i are not both null values, $p_1 \leqslant n_1$. These p_1 dimension similarity coefficients will be considered *valid*.

The *space similarity coefficient ss*, of two entities a, b over a concern space S_1 is the average of the p_1 *valid* similarity coefficients out of the n_1 dimensions in the concern space S_1.

$$ss_1 = \frac{\sum_{e=1}^{p_1} ds_e}{p_1} \tag{3}$$

The *multispace similarity coefficient*, ms, of two entities a, b over a set of k concern spaces $S_1, ..., S_k$ is the average of the space similarity coefficients of all k spaces in the concern spaces selection with respect to the number of

dimensions.

$$ms = \frac{\sum_{j=1}^{k} ss_j}{k} \tag{4}$$

All these coefficients: dimension similarity (2), space similarity (3), multi-space similarity (4), are taking subunitary values in the range of [0,1], where 0(zero) represents the weakest similarity for two entities (no similarities), while 1 represents the strongest similarity (identical entity concerns).

To conclude this section, the main advantage of the MultiCoS approach is tracing entities with *similar* concerns, and the ability to search entities that are matching specific concern profiles in a certain measureable level.

4. Case study MultiCoS

A Case Study using the MultiCoS approach was conducted. The scenario is that a web application had to be developed. Beside the regular resources provided in order to develop the product, a web application already developed is available with all its artefacts marked up using the MultiCoS approach. This means the existing web application has at least three system spaces (Requirements Space, Documents Space, Code Space) connected to each other by conceptual bindings, based on their relations to the Concern Spaces in the application. The goal is to reuse documents and code from the existing web application to support the development of the new one.

Being established this, we will consider two different web applications, Application A, already developed, called *Read-eng*, an ecommerce web-application selling books, and, Application B, required to be developed, *Target-Ear*, a multimedia web-application for listening music online.

In the Application A, Read-eng, the user can search and view books in store, he can add/remove books in the shopping cart, and he can start the process of buying the books if he is logged in. In the Application B, Target-Ear, the user can search and listen to individual song, he can add/remove song in a playlist, and he can play the playlist only if he is logged in.

The applications are quite different, from services provided, to the potential customers. Still, there are certain functionalities we can relate in the two applications. So, users in both applications can search for content, can open/view an *item* (book/song), can manage a list of items, and they both can use the items in the list for the intended purpose in case they are logged.

In the following, the functionalities of the two applications are written as requirements. The requirements in each application will form a separate Requirements Spaces. Using their value of the f function in the concern spaces, we will relate the requirements in the two applications, establishing

conceptual bindings between certain requirements. When such relations are being established, using a similar process, relations can be created between the entities from the Document Space and from the Code Space in the two applications. The way the requirements are set in the Table 1 gives the reader the impression that requirements are similar. We will check if requirements are similar using MultiCoS aproach.

TABLE 1. The Requirements Spaces of the applications A and B

Aplication A Requirements Space	
id	**Requirement**
RA1	Search book
RA2	View book
RA3	**Add book to shopping cart**
RA4	Remove book from shopping cart
RA5	Login
RA6	Logout
RA7	Create account
RA8	**Logged user can buy items in shopping cart**

Application B Req. Sp.	
id	**Requirement**
RB1	Search song
RB2	Listen song
RB3	**Add song to playlist**
RB4	Remove song from playlist
RB5	Login
RB6	Logout
RB7	Create account
RB8	**Logged user can listen songs in playlist**

Some concern spaces considered for this case study are meant to describe standard properties of different entity types in any type of software systems, others are specific to web applications.

In the last column of Table 2 we notice that Concern Spaces can be included in multispaces collections based on the type of entities they can concern. Concern Spaces 1, 2, 3, 4, 9, 10, 11 can be applied on entities of r type ($r = requirements$), Concern Spaces 1, 2, 4, 5, 7, 8, 10 can be applied on entities of c type ($c = code$).

Knowing this, two requirements, or two snippets of code can be searched for being *similar*. Considering requirement RA8 from application A and requirement RB8 from application B, we will establish the value of f function for each one of them in the concern spaces and we will calculate the space similarity coefficient for each space and the multispace similarity coefficient.

As we can see, in Table 3, the multispace similarity coefficient for RA8 and RB8 is not very high, just 0.627. This happens due to low *space similarity coefficients* on concern spaces S9, S10, S11, S4, as the two requirements do concern different activities (buy vs. play), they are involved differently in the MVC context, they are linked differently to the NFR, and they involve different actions to the DB. Still, given the high values of $ss1$ and $ss2$, some artefacts used in the process are *similar* and can be reused.

A different kind of situation occurs considering the *code* that implements RA3 and RB3.

TABLE 2. Samples of Concern Spaces used in MultiCoS

id	Concern Space:m*	Concern Dimensions	for
S1	**Type of user:1**	Admin, superuser,user, guest	r,c**
S2	**Info Source:1**	Database, user, hostmachine, application	r,c
S3	**Prioritization:10**	Value to Customer, Value to Business, Implementation Cost, Ease of Implementation, Time to Implement	r
S4	**DB Action type:1**	Create, Read, Update, Delete, Static	r,c
S5	**Navigation type:1**	Get, Post, Header, Hyperlink, State	c
S6	**App type:10**	Integrated system, Eshop, Multimedia, Ubiquitos, Social	t,h
S7	**Serv. side lang.:1**	Python, Php, ASP, Ruby, Perl, .NET, C#	c
S8	**Data format:1**	DB, text, XML, UML, HTML, CSV, richtext, DOM, xls	c,t,a
S9	**Activity details:1**	add to list, remove from list, view details, buys, plays, logs	r,c,u
S10	**MVC:10**	Model, View, Controller	r,c,t
S11	**NFR:10**	sleekdesign, loadspeed, volatility, security	r,t

*m=max_value

**r=requirements, c=code, t=technology, a=artefacts, u=user, h=architecture

TABLE 3. The multispace similarity coefficient for requirements RA8 and RB8

Concern Space	Requirement RA8	Requirement RB8	Space similarity coefficient
S1	f(RA8,S1) = (0,1,1,0)	f(RB8,S1) = (0,1,1,0)	ss1 = 1
S2	f(RA8,S2) = (0,1,0,1)	f(RB8,S2) = (0,1,0,1)	ss2 = 1
S3	f(RA8,S3) = (5,10,9,9,8)	f(RB8,S3) = (10,9,9,8,9)	ss3 = 0.84
S4	f(RA8,S4) = (1,0,1,0,0)	f(RB8,S4) = (1,1,0,0,0)	ss4 = 0.33
S9	f(RA8,S9) = (0,0,0,1,0,1)	f(RB8,S9) = (0,0,1,0,1,1)	ss9 = 0.25
S10	f(RA8,S10) = (8,0,10)	f(RB8,S10) = (5,8,10)	ss10 = 0.37
S11	f(RA8,S11) = (7,5,2,10)	f(RB8,S11) = (9,10,6,5)	ss11 = 0.60
		multispace similarity	**ms=0.627**

The high *multispace similarity coefficient* obtained in Table 4, indicates that the code implementing requirement RA3 in application A: "*add book to shopping cart*" can be reused to implement the requirement RB3 in application B: "*add song to playlist"*. The high value of the coefficient calculated indicates

TABLE 4. The multispace similarity coefficient for code implementing requirements RA3 and RB3

Concern Space	**code for req. RA3**	**code for req. RB3**	**Space similarity coefficient**
S1	f(RA3,S1)=(0,1,1,1)	f(RB3,S1)=(0,1,1,1)	ss1 = 1
S2	f(RA3,S2)=(1,0,0,1)	f(RB3,S2)=(0,1,0,1)	ss2 = 1
S4	f(RA3,S4)=(0,1,1,0,0)	f(RB3,S4)=(0,1,1,0,0)	ss4 = 1
S5	f(RA3,S5)=(0,1,1,0,0)	f(RB3,S5)=(0,1,1,0,0)	ss5 = 1
S7	f(RA3,S7)=(0,1,0,0,0,0,0)	f(RB3,S7)=(0,1,0,0,0,0,0)	ss7 = 1
S8	f(RA3,S8)=(1,0,1,0,1,0,0,0,0)	f(RB3,S8)=(1,0,1,0,1,0,0,1,0)	ss8 = 0.75
S9	f(RA3,S9)=(1,0,0,0,0,0)	f(RB3,S9)=(1,0,0,0,0,0)	ss9 = 1
S10	f(RA3,S10)=(8,5,7)	f(RB3,S10)=(6,5,6)	ss10 = 0.9
		multispace similarity	**ms=0.969**

that the two snippets of code have the same semantic, even they are not identical, and the applications they serve are quite different. Given this, the code of the application A can be reused, with small adjustments, in developing a new one, the application B.

5. Further work and conclusions

The current approach and the metric presented can be extended to calculate similarity coefficients for applications and for different types of systems. A framework under development have to be completed to support this approach. Investigating and determining proper concern spaces can extend the work of Poshyvanyk et all in [13]. One goal of the approach is to add semantic to the entities using Concern Spaces. The process of mapping entities in the concern spaces using the function f is a non-automatic process, performed by a person, based on his/her considerations, and might be different from one person to another one. As the similarity coefficients are calculated based on these mappings they can have slightly different value. The way the coefficients are calculated and the fact that their values are in the [0,1] range can relate the work presented here to fuzzy mathematics.

Acknowledgements: This work was possible with the financial support of the Sectoral Operational Programme for Human Resources Development 2007-2013, co-financed by the European Social Fund, under the project number POSDRU/107/1.5/S/76841 with the title Modern Doctoral Studies: Internationalization and Interdisciplinarity.

References

[1] Castro J., Kolp M., and Mylopoulos J. Towards requirements-driven information systems engineering: the Tropos project - *Information Systems 27* (2002) 365-389.
[2] Chen X.., Liu Z.., Mencl V., Separation of Concerns and Consistent Integration in Requirements Modelling. Macao, China, 2007.
[3] Dijkstra, E. A Discipline of Programming. 0-13-215871-X. Prentice-Hall 1976, pp 15-25.
[4] Gal-Chis C.E.N., A Multi-Dimensional Separation of Concerns of the Web Application Requirements, *Studia Universitatis Babes-Bolyai, Inf.*, V. LVIII, nr. 3, 2013, pp 29-40.
[5] Gal-Chis C.E.N., Modeling Concern Spaces Using Multi Dimensional Separation of Concern *International Journal of Computers and Techology Vol 11, No 2*: IJCT 2013, pp 2302-2313.
[6] William Harrison W., Ossher H., Tarr P. General Composition of Software Artifacts, *Proceedings of Software Composition Workshop 2006*, March 2006, Springer-Verlag, LNCS 4089, pp 194-210.
[7] Jacobson I., Ng P.W., *Aspect-Oriented Software Development with Use Cases*, Add.-W., 2004.
[8] Jendrik J., Uwe A. Concern-Based (de)composition of Model-Driven Software Development Processes, *Model Driven Engineering Languages and Systems* 2010 pp 47-62.
[9] Kaminski P. Applying Multi-dimensional Separation of Concerns to Software Visualization - *Workshop on Advanced Separation of Concerns*, ICSE 2001.
[10] Kiczales, G., J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J.-M. Loingtier, J. Irwin. Aspect-Oriented Programming, in: *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP 1997)*, Jyvskyl, Finland, Lecture Notes in Computer Science 1241, Springer-Verlag, pp 220-242.
[11] Moreira A., Rashid A., Arajo J., Modularization and Composition of Aspectual Requirements, in *2nd Aspect-Oriented Software Development Conf.*, Boston, USA, 2003.
[12] Moreira A., Rashid A., Arajo J., Multi-Dimensional SoC in RE. *IEEE*, 2005.
[13] Poshyvanyk, D., Gethers, M., and Marcus, A., Concept Location using Formal Concept Analysis and Information Retrieval, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21(4), 2012.
[14] Ossher H., Tarr P. , MultiDimensional SoC and the Hyperspace Approach, 2002.
[15] http://trese.cs.utwente.nl/taosad/separation_of_concerns.htm.
[16] Sutton Jr., S. M., Rouvellou, I. Modeling of Software Concerns in Cosmos. *1st International Conference on Aspect-Oriented Software Development*, Enschede, NL, April, 2002.
[17] Tarr, P., H. Ossher, W. Harrison, S.M. Sutton Jr. (1999): N Degrees of Separation: Multi- Dimensional Separation of Concerns, in: *Proceedings of the 21st International Conference on Software Engineering*, Los Angeles, California, USA, IEEE Computer Society Press, pp107-119.

Faculty of Mathematics and Computer Science, Babes-Bolyai University, Cluj-Napoca

E-mail address: calin.gal-chis@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

EXTENDING OBJECT-RELATIONAL MAPPING WITH CHANGE DATA CAPTURE

RADOSLAV RADEV

Abstract. The most critical disadvantage of the Change Data Capture functionality provided by the relational database management systems is the impossibility to add custom, application-specific data to the log. In this paper an approach is proposed how to implement custom, application-specific Change Data Capture functionality with the help of database triggers and O/RM framework. There are listed clear problem requirements and concrete steps are proposed how to resolve it. The two layers of the solution are identified and described database level and application level. All details of the solution are provided with example source code. It is also suggested an approach for quality assurance of the solution.

1. Introduction

One of the open issues of object-oriented programming in the last two decades is the object-relational impedance mismatch [13], [7]. It concerns the usage of relational database management systems (RDBMS) with object-oriented programming languages. In 2002 Fowler summarized the concept of object-relational mapping (O/RM) [3] and in the last years several O/RM frameworks were developed, thus bringing object-relational technology to the mainstream of application programming [10]. Among them most known and used are Hibernate (for Java) [18] and Microsoft Entity Framework (for .NET) [11]. They are based on the entity data model conception and definition of mappings between object-oriented classes and relational database tables [6, 18].

Beside the latest development of object-oriented and other non-relational database systems, the relational databases are still widely used for enterprise and data-driven applications. This is due mostly to their simplicity, proven

Received by the editors: October 23, 2014.

2010 *Mathematics Subject Classification.* 68P05, 68P20.

1998 *CR Categories and Descriptors.* D.2.12 [**Software Engineering**]: Interoperability - *Data mapping*; H.2.4 [**Database Management**]: Systems - *Relational databases*.

Key words and phrases. relational database, change data capture, log-trigger, object-relational mapping.

reliability for almost 40 years of development and support by big software vendors like Oracle, Microsoft and IBM [3].

In 2002 Oracle introduced a new relational database functionality called Change Data Capture [20]. It identifies and captures data that has been added to, updated or removed from, relational tables, and makes the change data available for use by applications. This allows all changes made to the database records to be tracked, persisted and retrieved later. This is very useful for building statistics, reviewing logs or revert the changes made. In the last case the records affected could be restored to their previous state, because the Change Data Capture feature keeps the changes made to the records along with their previous versions.

In 2007 Popfinger gives a more precise definition of Change Data Capture architecture [22]. This architecture uses rules and triggers to identify data that has been changed since the last extraction. Triggers copy updated data to a specially created change table (log-table) where they can be accessed by subscribers using individual views to access the information it is interested in and allowed to read.

In 2008 Microsoft also implemented Change Data Capture in Microsoft SQL Server 2008 [15]. Change Data Capture functionality is very useful for tracking and keeping the changes made to database records. The whole process is actually very easy to use, because all the tracking and persistence of the changes is made by the database server itself (of course we assume it supports this functionality like in the examples above). But the following problem arises: we cannot associate additional, custom data with the changes made. Let us give one very common example. Imagine an enterprise or data-driven application [3], [12] that uses a relational database as a persistent storage. It allows different users to log in to the system and manipulate the data via user interface. The users can work simultaneously and perform different operations that result in changes of the database records. But with the default Change Data Capture functionality provided by RDBMSs we cannot distinguish which change is made by which user.

Why this is important? Because if we find a way to attach a user identifier to the Change Data Capture logs, we not only will be able to distinguish which change is made by which user, but also to revert the changes of a particular user without affecting the other users changes.

In this paper we propose an approach how to implement a custom Change Data Capture functionality that supports logging not only of the database record changes, but also of additional, custom, application-specific data for every record. To solve the upper problem we will use O/RM framework. We will centralize all the access to the relational database through the O/RM framework and will ensure that the additional data is saved correctly to the

Change Data Capture logs. In advance, our approach is not tied to a particular O/RM framework or a RDBMS, but suggests a general solution that could be implemented for different O/RM frameworks and RDBMSs according to their specific features. Finally, we will illustrate one practical application of our approach as a proof of concept - how to use this extended Change Data Capture functionality to revert a set of changes made to the database by a particular user in an enterprise application. In this way, we are trying to solve some current problems in database usage.

It has to note, that Change Data Capture technique is extensively used in other areas of computer science and is of particular importance for data warehouse maintenance (see [20], [8]), which is beyond of the scope of this study.

2. Requirements

Here we will specify in more details the problem and the requirement for its solution. We seek to define a general approach how to:

- Implement our own Change Data Capture functionality because the default one provided by RDBMSs (Oracle, Microsoft, etc.) cannot be easily extended to support custom data added to the log. That means we have to define a mechanism how to record all changes made to the database records and how to persist these changes.
- Along with the database records changes we want to persist also additional, application-specific data to the log for every changed record.
- The approach should not be tied to any particular O/RM framework or a RDBMS. It must suggest a general solution that could be implemented for different O/M frameworks and RDBMSs.
- Ensure a possibility for quality assurance of our solution. That means a way to ensure all preconditions in the database (tables, triggers, etc.) are satisfied and so our solution is guaranteed to work correctly.
- As prove of concept, revert a set of changes by a given criteria without affecting all other changes. For example, reverting only changes made in an enterprise application by a particular user, but keep all other changes made meanwhile by other users.

Our approach consists of concepts applied to two levels database level (RDBMS) and application level (O/RM). We will discuss the database level in Section 3 and the application level in Section 4.

3. Change Data Capture - Database Level

Let us call every table we want to track changes of data-table. We will create a common log-table to store the logs needed for Change Data Capture

functionality. This means we will store all the logs in a single log-table, not in a separate log-table for every data-table. For every change made to a record in any data-table in the database we will insert a log-record in the log-table with information about the change. This information consists of:

- Unique identifier of the operation.
- Table identifier (in which data-table the changes is made).
- Date and time of the operation (when the change is made).
- Old record data (record values before the change). In case of insert operation, it will be empty (NULL).
- New record data (record values after the change). In case of delete operation, it will be empty (NULL).
- Type of the operation (insert/update/delete) – optional, because it could be figured out from the previous two columns. If old record data is NULL, it is insert operation. If new record data is NULL, it is delete operation. If both are NOT NULL, it is update operation.
- Transaction identifier (needed by the O/RM framework to add custom additional data later).
- Additional data – it could be a single column, but if more complex additional data is to be persisted, it could be a single column in XML format or multiple different columns.

The old and new record data will be serialized in a common format. It is most convenient to use XML because some RDBMSs support serializing a record to XML, for example Oracle [21] and Microsoft [16]. If that is not the case with a particular RDBMS, another specific solution should be applied according to the functionalities provided by this specific RDBMS for record serialization.

Here is the DDL code for creating the log-table in Transact-SQL [4] for Microsoft SQL Server:

```
CREATE TABLE [dbo].[Log_Records]
(
    [ID] [uniqueidentifier] NOT NULL PRIMARY KEY DEFAULT
NEWSEQUENTIALID(), – Unique identifier of the operation – automatically generated by RDMBS.
    [Old_Data] [xml] NULL, – Old recor values serialized in XML format.
    [New_Data] [xml] NULL, – New record values serialized in XML format.
    [Table_Name] [nvarchar](50) NOT NULL, – Data-table name of the changed record.
    [Operation] [varchar](10) NOT NULL, – Code of the operation. For clearance we will use
'INSERT', 'UPDATE' or 'DELETE' string values. It could equally well be an integer value, for
example 1 (insert), 2 (update), 3 (delete).
    [Transaction_ID] [bigint] NOT NULL, – Needed by O/RM framework to add additional data to
the log-record later.
    [User_ID] [uniqueidentifier] NULL, – Custom, additional, application-specific data added to the
Change Data Capture logs.
```

```
    [Date_Time] [datetime] NOT NULL DEFAULT GETDATE() – Date and time of the operation,
by default the current datetime.
)
```

To capture the changes made to the records we propose to use database triggers. They are routines executed before, instead, or after an event in a database [14]. In our case this event is an insert, update, or delete operation in a data-table. Triggers could be used for different purposes logging, validation, ensuring data integrity, or others. Another aim could be introduction of some business logic to the database [9]. In our case we use database triggers for the logging process of changes [1] and also for including some business-specific data to the log.

We will add log-trigger to every table for which we want to track the changes. It will be executed after every insert, update or delete operation. If the particular RDBMS allows it, the code of the trigger could be the same for all data tables, which makes it highly maintainable and testable. This is possible for Oracle and Microsoft SQL Server, because they support serializing a record to XML. Then we will have multiple triggers, one per data-table, but with the same DDL code, which makes them easy to maintain.

Here is our suggestion for a DDL code that creates a log-trigger for a data-table in Microsoft SQL Server. T-SQL, also known as Transact-SQL, is a proprietary extension to SQL, central to using Microsoft SQL Server [4]. All SQL and DDL examples in this paper will be given in T-SQL.

```
CREATE TRIGGER [dbo].[TR_Persons_Log]
    ON [dbo].[Persons] – in this case the data-table is dbo.Persons
    AFTER INSERT, UPDATE, DELETE
    AS
BEGIN
    – Serialize the old record values to a XML variable.
    DECLARE @old_data XML;
    SELECT @old_data = (SELECT * FROM deleted FOR XML PATH(''));
    /* Make notice that we do not use the column names of the data-table, which proves that
the log-trigger DDL code could be the same for all tables, because it does not depend on the table
metadata. */
    – Serialize the new record values to a XML variable.
    DECLARE @new_data XML;
    SELECT @new_data = (SELECT * FROM inserted FOR XML PATH(''));
    – Fill the operation column value as user-friendly text for clearance.
    DECLARE @operation VARCHAR(10);
    SELECT @operation = CASE
    WHEN (@old_data IS NULL) and (@new_data IS NOT NULL) THEN 'INSERT'
    WHEN (@old_data IS NOT NULL) and (@new_data IS NOT NULL) THEN 'UPDATE'
    WHEN (@old_data IS NOT NULL) and (@new_data IS NULL) THEN 'DELETE'
    END;
    – If both old and new data is null  nothing to do.
```

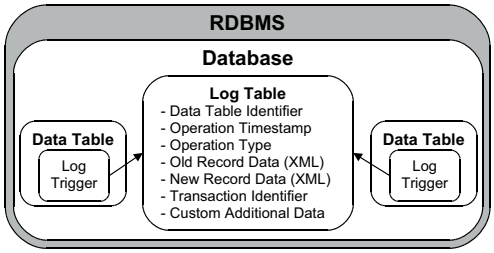


FIGURE 1. Scheme of Change Data Capture - Database level

```
    IF (@old_data IS NULL and @new_data IS NULL) RETURN;
     – Retrieve the current transaction identifier  this is needed by O/RM to add additional data
later.
    DECLARE @transaction_id INT;
    SELECT @transaction_id = transaction_id
    FROM sys.dm_tran_current_transaction;
     – Get the current table name in format [schema].[table].
    DECLARE @table_name NVARCHAR(50);
    SELECT @table_name = s.name + '.' + t.name
    FROM sysobjects tr
    INNER JOIN sys.tables t ON tr.parent_obj = t.object_id
    INNER JOIN sys.schemas s ON s.schema_id = t.schema_id
    WHERE tr.id = @@procid;
     – Insert the new log-record in the log-table.
    INSERT INTO dbo.Log_Records (Old_Data, New_Data, Operation, Table_Name, Transaction_ID)
    VALUES (@old_data, @new_data, @operation, @table_name, @transaction_id);
END
```

All these operations proposed by us are implemented on the database server level, as shown on Figure 1. Make notice that so far we have implemented only the basic Change Data Capture functionality that tracks the changes made to the records. No additional data has been added to the log yet, only the storage for it has been prepared in the common log-table.

4. Change Data Capture with OR/M Framework - Application Level

On application level we will use O/RM framework to add custom additional data to the log. O/RM framework centralizes all database access from within the software application. It both separates and bridges the object-oriented classes or entities and the relational database.

O/RM frameworks frequently use Unit of Work design pattern, defined by Fowler [3]. This is the case for example with Hibernate [5] and Entity Framework [2]. Unit of Work design pattern ensures that all operations made to different entities are submitted to the database within a single transaction.

We will use this specific feature to encapsulate both logging the changes and adding the additional data to the log in this single transaction, as shown in Figure 2.

We propose the following sequence of operations made by O/RM framework to support Change Data Capture functionality and to add additional data to the log:

(1) Client/User manipulates the entities via O/RM framework.
(2) Client/User calls O/RM frameworks *SaveChanges* method and passes to it as parameters the additional data that is to be saved to the log.
(3) O/RM framework opens a connection to the database, starts a transaction and retrieves the transaction identifier from the database.
(4) O/RM framework generates SQL statements corresponding to the changes made to its entities and sends them to the database server.
(5) Database server executes these SQL statements consequently. Because every table has a log-trigger that is executed after every insert, update or delete operation, the log-trigger is fired for every inserted/updated/deleted record in any table we want to track changes of.
(6) On every execution, the log-trigger inserts a single record in the log-table (log-record) with the following data: timestamp, table identifier, type of the operation, old record data, new record data, transaction identifier.
(7) O/RM framework add the additional data to the log by generating and executing a SQL statement to update all log-records with the transaction identifier retrieved in step 3. This ensures that only the newly inserted log-records (in previous step 6) will be updated, so any other changes made simultaneously by other users/clients will not be affected.
(8) O/RM framework commits the transaction started in step 4 and closes the connection to the database.

Here is our suggestion for a commit method of the O/RM framework. It is given in C# for Entity Framework 4.0 [11].

```
/* The Object Context class  instance of the main O/RM framework class. In case of Entity Framework, this is the ObjectContext class, so we inherit it. */
public partial class DbRollbackContext : ObjectContext
{
    /* Pass the additional data that is to be saved to the log as parameters of the SaveChanges method  in our case, only the current users identifier, but there could be also other parameters. */
    public void SaveChanges(Guid userId)
    {
    this.Connection.Open(); // Open a connection to the database.
    try
```

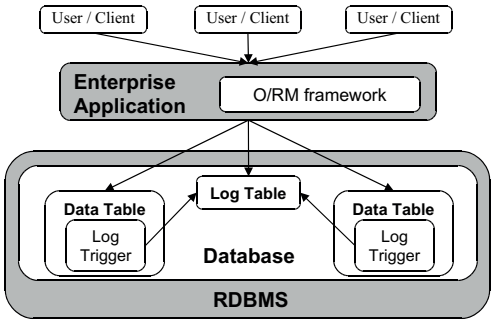


FIGURE 2. Scheme of Change Data Capture - Application level.

```
    {
     // Start a transaction to the database within the opened connection.
    using (DbTransaction transaction = this.Connection.BeginTransaction())
    {
    /* Retrieve the just started transaction identifier. This is the same value that will be saved by
the log-trigger in the Transaction_ID column of the log-table for all log-records that will be inserted
for the changes that are to be made in the next line of code. */
    long transationId = this.ExecuteStoreQuery<long>("SELECT
    transaction_id FROM sys.dm_tran_current_transaction").First();
    /* Call the base SaveChanges method of the ObjectContext class. It will generate and execute
SQL statements corresponding to the changes made to the entities tracked by the object context.
These SQL statements are insert/update/delete operations that will fire the log-trigger of the cor-
responding tables. The log trigger will insert log-records in the log-table with the current Transac-
tion_ID, which was retrieved in the previous line of code. */
    base.SaveChanges();
    /* Update all log-records with the same Transaction_ID, i.e. the just inserted ones  and set
their additional data. */
    this.ExecuteStoreCommand(string.Format("UPDATE dbo.Log_Records SET User_ID = '0'
    WHERE User_ID IS NULL AND Transaction_ID = 1", userId, transationId));
    transaction.Commit(); // Commit the transaction.
    }
    }
    finally
    {
    this.Connection.Close(); // Close the connection to the database.
    }
    }

}
```

With this the implementation of our custom Change Data Capture functionality is complete. In the next Section we will discuss how to retrieve the inserted logs and how to use them to revert a set of operations.

5. An Example of Practical Application - Retrieve the Logs and Revert a Set of Changes

Once the log-records are in the log-table, it is easy to retrieve them with a simple SELECT SQL command. And because we have also the additional data stored in the log, we can filter the log-records accordingly. For example, we can retrieve the set of changes made by a user for a time period. Moreover, we can easily revert these changes. We have the order of the changes in the *Date_Time* column. For every log-record we can construct a corresponding revert SQL statement. Then we must execute these revert statements in the opposite order.

5.1. **Revert INSERT Operation.** To revert an INSERT operation, we will generate a corresponding DELETE statement. In the WHERE clause of the DELETE statement we will include all old record values (before the change logged by the log-record we want to revert). This ensures that the newly generated DELETE statement will actually delete the record only if the current record values are the same as the original ones in the moment of its insertion. That is to say, if the record has been updated later by another user, its current record values will be different from the original ones. The WHERE clause of the DELETE statement will match no record, and accordingly no records will be deleted. So if another user has changed the record after that, our newly generated DELETE statement will not affect them actually it will not affect any records. What to do in this case is a decision that depends of the business logic of the enterprise application. It could regard such situation as an error or not; it could stop the whole process of reverting or to try to continue it. Different approaches are possible according to the concrete situations.

Here is an example code how to generate a DELETE statement that will revert an INSERT statement from a log-record.

```
private static string GenerateRevertInsertSqlStatement(LogRecord logRecord)
{
    /*Convert the old data XML to Dictionary with key the column name and value - the old
record value for this column (before the changed logged in this logged record). */
    IDictionary<string, string> newRecordValues =
    ConvertXmlToDictionary(logRecord.NewData);
    /*Delete the inserted record, but check if it has not been updated later. For the purpose include
the record values in the moment of insertion in the WHERE clause. This guarantees the record will
not be deleted if it has been updated later by another user.*/
    return string.Format("DELETE FROM {0} WHERE {1}", logRecord.TableName,
    string.Join(" AND ", newRecordValues.Select(v =>
    string.Format("{0} = '{1}'", v.Key, v.Value))));
}
```

5.2. **Revert UPDATE Operation.** To revert an UPDATE operation, we will generate another UPDATE SQL statement, taking the following guidelines into consideration:

- In the SET clause of the UPDATE statement, we will update only the changed columns values.
- If the records table contains computed column [5], they should not be included in the SET clause, because they will be computed by the RDBMS. Information about the computed columns is usually available in the O/RM framework meta-data.
- In the WHERE clause of the UPDATE statement we will include all old record values (before the change logged by the log-record we want to revert). This ensures that the newly generated UPDATE statement will actually update the record only if the current record values are the same as in the moment of inserting the log-record. That is to say, if the record has been updated again later by another user, its current record values will be different from the ones in the log-record. The WHERE clause of the UPDATE statement will match no record, and accordingly no records will be updated. So if another user has changed the record after that, our newly generated UPDATE statement will not affect them actually it will not affect any records. Again, what to do in this case is a decision that depends of the business logic of the enterprise application, as stated in Section 5.1.

Here is an example code how to generate an UPDATE statement that will revert an UPDATE statement from a log-record.

```
private static string GenerateRevertUpdateSqlStatement(LogRecord logRecord)
{
    /* Convert the old data XML to Dictionary with key the column name and value - the old
record value for this column (before the changed logged in this logged record). */
    IDictionary<string, string> oldRecordValues = ConvertXmlToDictionary(logRecord.OldData);
    /* Convert the new data XML to Dictionary with key the column name and value - the new
record value for this column (after the changed logged in this logged record). */
    IDictionary<string, string> newRecordValues = ConvertXmlToDictionary(logRecord.NewData);
    StringBuilder setStatement = new StringBuilder();
    StringBuilder whereStatement = new StringBuilder();
    /* If the table contains computed columns, they should be excluded from the SQL statement.
These special columns can be retrieved from O/RM mappings meta-data. */
    // Iterate through the new record values to build the UPDATE SQL statement.
    foreach (KeyValuePair<string, string> newRecordValue in newRecordValues)
    {
    string oldRecordValue = oldRecordValues[newRecordValue.Key];
     // If the new value is the same as the old one, do not include it in the SET clause.
    if (oldRecordValue ! = newRecordValue.Value)
    {
    if (setStatement.Length > 0) setStatement.Append(", ");
```

```
    setStatement.AppendFormat("0= '1'", newRecordValue.Key, oldRecordValue);
    }
    /* Include all old values in the WHERE clause to guarantee that the record will be updated
only if the current record values match the "new" ones (after the change). */
    if (whereStatement.Length > 0) whereStatement.Append(" AND ");
    whereStatement.AppendFormat("0 = '1'", newRecordValue.Key, newRecordValue.Value);
    }
    // No fields have been changed - nothing to revert.
    if (setStatement.Length == 0) return string.Empty;
    // Concatenate and return the whole SQL statement.
    return string.Format("UPDATE 0 SET 1 WHERE 2", logRecord.TableName, setStatement,
whereStatement);
}
```

5.3. **Revert DELETE Operation.** To revert a DELETE operation, we will generate INSERT SQL statement to re-insert the deleted record. In order to achieve this we have to take the following into consideration:

- If the records table contains computed column [24], they should not be included in the SET clause, because they will be computed by the RDBMS. Information about the computed columns is usually available in the O/RM framework meta-data.
- If the records table contains auto-generated columns [17], some RDBMSs do now allow inserting concrete values (in our case, the old ones in the moment of records deletion). An example of such a restriction is Microsoft SQL Server [17]. In this case the RDBMS must be explicitly instructed to allow insertion of auto-generated values with DDL statement (if it supports it). Because many inserts may be needed in the whole process of reverting records, it is best to execute these enable auto-generated values DDLs against the tables needed in the beginning of the whole revert process and undo them (i.e. restore the restriction of inserting auto-generated values) in the end of the whole revert process, after all SQL statements generated by the log-records have been executed. Example code will be shown in the next Section 5.4.

Here is an example code how to generate an INSERT statement that will revert a DELETE statement from a log-record.

```
private static string GenerateRevertDeleteSqlStatement(LogRecord logRecord)
{
    IDictionary<string, string> oldRecordValues = ConvertXmlToDictionary(logRecord.OldData);
    /* Re-insert the deleted record with its last values. If the table contains computed columns, they
should be excluded from the SQL statement. These special columns can be retrieved from O/RM
mappings meta-data. */
    return string.Format("INSERT INTO 0 (1) VALUES (2)", logRecord.TableName,
```

```
    string.Join(", ", oldDataValues.Select(v => v.Key)),
    string.Join(", ", oldDataValues.Select(v => string.Format("'0'", v.Value))));
}
```

5.4. **O/RM framework Revert Changes Method.** The O/RM framework Revert Changes method should execute the following operations:

(1) Retrieve a set of log-records filtered by a given criteria, for example for specific user and/or time interval.
(2) Identify the data-tables that need structural changes because of auto-generated columns, as explained in Section 5.3.
(3) Open a connection to the database.
(4) Generate and execute DDL statements for the tables identified in Step 2. to enable insertion of concrete values in auto-generated columns.
(5) Starts a transaction to the database all revert operations should be in a transaction.
(6) For every log-record generate and execute a revert SQL statement. If the revert SQL statement has not affected any records, handle the situation that the record we try to revert the changes of has been changed later, for example by another user. It is up to the enterprise application how to handle this situation. At least a few approaches are possible to abort the whole revert process; to ignore the situation and to continue with the next changes and to try to revert them; or to require a user interaction how to proceed
(7) Commit the transaction to the database.
(8) Generate and execute DDL statements for the tables identified in Step 2. to disable insertion of concrete values in auto-generated columns, i.e. to return the tables to their initial state.
(9) Close the connection to the database.

Here is an example revert changes method in C# for Entity Framework 4.0 [5].

```
public void RevertChanges(Guid userId, DateTime startDateTime, DateTime endDateTime)
{
    // Get the logs filtered by the given criteria.
    IEnumerable<LogRecord> logsToRevert = this.LogRecords
    .Where(r => (r.User_ID == userId) &&
    (r.Date_Time >= startDateTime) &&
    (r.Date_Time <= endDateTime))
    .OrderByDescending(r => r.Date_Time)
    .ToArray();
    /* Retrieve the tables with deleted records because the revert operation is an INSERT operation,
we need to enable insert of identity/auto-generated values in this tables. */
    IEnumerable<string> tablesWithDeletedRecordsThatWillBeReinserted = logsToRevert
```

```
.OfType<DeleteLogRecord> ()
.Select(r => r.Table_Name)
.Distinct()
.ToArray();
this.Connection.Open(); // Open a connection to the database.
try
{
 /* Enable insert of identity/auto-generated values for each table with deleted records that will
be re-inserted. */
foreach (string tableName in tablesWithDeletedRecordsThatWillBeReinserted)
{ string cmdEnableIdentityInsertSql = string.Format("SET IDENTITY_INSERT 0 ON", table-
Name);
this.ExecuteStoreCommand(cmdEnableIdentityInsertSql); }
 // Start a transaction to the database within the opened connection.
using (DbTransaction transaction = this.Connection.BeginTransaction())
{
foreach (LogRecord logRecord in logsToRevert)
{ string revertSql = null;
 // Generate the revert SQL statement according to the operation.
switch (logRecord.Operation)
{
case "INSERT": revertSql = GenerateRevertInsertSqlStatement(logRecord); break;
case "UPDATE": revertSql = GenerateRevertUpdateSqlStatement(logRecord); break;
case "DELETE": revertSql = GenerateRevertDeleteSqlStatement(logRecord); break;
}
 // UPDATE statement that has made no changes - ignore it.
if (string.IsNullOrEmpty(revertSql)) continue;
try
{ // Execute the command against the database.
int recordsAffected = this.ExecuteStoreCommand(revertSql);
 // Check the result
if (recordsAffected == 0)
{
/* The revert statement has not affected any records, i.e. its WHERE clause is not matched
by any record in the database. This means that the record that is to be reverted has been changed
by another user after the change. Usually an exception should be thrown here, but a more complex
logic is possible according to the enterprise application specific needs. */
}
}
catch (Exception ex)
{
 /* Executing revert SQL statement has failed, due to constraints violation or some other
reason. More detailed information about the error can be retrieved from ex variable. */
}
}
transaction.Commit(); // Commit the transaction.
}
}
finally
{
 /* Disable insert of identity/auto-generated values for each table with deleted records that have
been be re-inserted, i.e. return the table to its initial state. */
foreach (string tableName in tablesWithDeletedRecordsThatWillBeReinserted)
```

```
    {
    string cmdDisableIdentityInsertSql = string.Format("SET IDENTITY_INSERT 0 OFF", table-
Name);
    this.ExecuteStoreCommand(cmdDisableIdentityInsertSql);
    }
    this.Connection.Close(); // Close the connection to the database.
    }
}
```

---

6. Quality Assurance

One of the fundamental concerns in software engineering is Quality Assurance (QA) [23]. Very common methodology for its realization is the unit tests [19]. Here we will describe how to implement unit tests to ensure the correct behavior of our solution. We propose to test a few things:

(1) All data-tables in the database must have log-triggers.
(2) The code for every log-trigger should be correct. If we need to change the log-trigger code in time, and because the log-trigger code is basically the same for all data-tables, we must be sure that all data-tables have the latest version of the log-trigger. So as correct we will regard the latest version of the log-trigger code.

There exists various options for implementation of these unit tests. We will give an example with implementation in the following way: We will store the correct (i.e. the latest) log-trigger code as an embedded file in the assembly that contains the O/RM object context. The O/RM framework will read this file. The code of the log trigger is the same for all tables, except the data-table name. So the O/RM framework will store a template log-trigger code and replace it with the given table name.

The unit test must execute the following actions to ensure all data-tables in the database have a correct log-trigger:

(1) Retrieve all data-tables from the RDBMSs metadata with the code of their log-trigger.
(2) Check if they have a log-trigger at all.
(3) Check if the code of the log-trigger is correct.

If there are other triggers in the database, it will be good to have a naming convention for them in order to distinguish the log-triggers from other triggers. An example for a naming convention is TR_TABLE_NAME_Log, but it could equally well be any other. Also, if we do not want to track changes for all data-tables, but only for specific ones, we should modify our unit test to retrieve from the database only the data-tables we want to track the changes of.

7. Conclusion and Further Work

In this paper an approach is proposed how to implement custom Change Data Capture functionality with the help of O/RM framework. The clear problem requirements are listed along with the proposed steps for our solution divided in two layers database level (RDBMS) and application level (O/RM framework). Fully-working and well-documented example source code is given for every part of the solution along with an approach for quality assurance. We proved empirically the correctness of our approach implementing it successfully in a real-world business application for *interAxio* project in Healthcare Research & Development department of the Belgium company Televic. The following technologies were used: Microsoft .NET Framework 4.0, C# 4.0, Entity Framework 4.0 [5], SQL Server 2008 R2.

As future perspectives for our approach we consider: (1) Examine and prove empirically its application for other RDBMSs and O/RM frameworks; (2) Measure and evaluate the performance impact of the whole system where it is integrated; (3) Explore how this logging could interfere with systems that are using triggers for other things (pre-insert/post-insert triggers as example), and where would the log trigger be inserted in the chain of called triggers; (4) Does this procedure of logging could infere with other transactions and how?

References

[1] A.A. Chuvakin, K.J. Schmidt, Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management, Syngress, Waltham, 2012.

[2] Developer Network, DbContext class documentation, Microsoft, http://msdn.microsoft.com/en-us/library/system.data.entity.dbcontext

[3] M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley Longman Publishing Co. Inc., Boston, 2002.

[4] K. Henderson, The Guru's Guide to Transact-SQL. Addison-Wesley Longman, Reading, MA, USA, 2000.

[5] Hibernate Community Documentation, Chapter 11. Transactions and Concurrency, http://docs.jboss.org/hibernate/core/3.3/reference/en/html/transactions.html, (2014).

[6] A.T.F. Hutt, Data mappings again, Software: Practice and Experience, 8(1978), pp. 483493.

[7] C. Ireland, D. Bowers, M. Newton, K. Waugh, A classification of object-relational impedance mismatch, in Proc. of First International Conference on Advances in Databases, Knowledge, and Data Applications, IEEE, 2009, pp. 36-43.

[8] R. Kimball, J. Caserta, The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data, John Wiley & Sons, Inc., 2004.

[9] G. Knolmayer, H. Herbst, M. Schlesinger, Enforcing Business Rules by the Application of Trigger Concepts, in Proc. of Priority Programme Informatics Research, Information Conference, Module 1, Swiss National Science Foundation, Berne, Switzerland, 1994, pp. 24-30.

[10] V. Krishnamurthy, S. Banerjee, A. Nori, Bringing object-relational technology to the mainstream, in Proc. of SIGMOD '99 ACM International Conference on Management of Data, ACM, New York, 1999, pp. 513-514.
[11] J. Lerman, Programming Entity Framework, Building Data Centric Apps with the ADO.NET Entity Framework, OReilly Media, Sebastopol, 2009.
[12] D.S. Linthicum, Enterprise Application Integration. Addison-Wesley Longman, Reading, MA, USA, 2000.
[13] O.L. Madsen, Open issues in object-oriented programming - A Scandinavian perspective, Software: Practice and Experience, 25(S4), (1995), pp. 343.
[14] J. Melton, A.R. Simon, SQL: 1999, Understanding Relational Language Components, Chapter 11 Active Databases and Triggers. Morgan Kaufmann, San Francisco, 2002.
[15] Microsoft MSDN documentation, Tracking Data Changes, Change Data Capture, http://msdn.microsoft.com/en-us/library/bb522489(v=sql.105).aspx, (2014).
[16] Microsoft SQL Server documentation, For XML (SQL Server), http://technet.microsoft.com/en-us/library/ms178107(v=sql.100).aspx, (2014).
[17] Microsoft SQL Server documentation, SET IDENTITY_INSERT (Transact-SQL), http://technet.microsoft.com/en-us/library/ms188059.aspx, (2014).
[18] E.J. O'Neil, Object/relational mapping 2008: hibernate and the entity data model, in Proc. of SIGMOD '08 ACM SIGMOD International Conference on Management of data, ACM, New York, 2008, pp. 1351-1356.
[19] M. Olan, Unit testing: test early, test often, J. Comput. Sci. in Colleges archive, 19(2003), pp. 319-328.
[20] Oracle9i Data Warehousing Guide, Release 2 (9.2), Part Number A96520-01. Oracle Corporation, 2002.
[21] Oracle9i Supplied PL/SQL Packages and Types Reference, Release 2 (9.2), Part Number A96612-01, Chapter 85: Functions and Procedures of DBMS_XMLGEN, Oracle Corporation, 2002.
[22] C. Popfinger, Enhanced active database for federated information systems, Doctoral Thesis/Dissertation, Heinrich-Heine-Universität Düsseldorf, GRIN Verlag, Düsseldorf, 2007.
[23] G. Schulmeyer, J.I. McManus, Handbook of Software Quality Assurance, Van Nostrand Reinhold Co., New York, 1987.
[24] N. Sharma, L. Perniu, R.F. Chong, A. Iyer, A.C. Mitea, C. Nandan, M. Nonvinkere, M. Danubianu, Database Fundamentals. IBM Corporation, 2010.

Department of Computer Systems, Faculty of Mathematics and Informatics, Plovdiv University 'Paisii Hilendarski', Tzar Asen str. 24, 4000 Plovdiv, Bulgaria

E-mail address: radoslav_radev@gbg.bg

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

ADAPTIVE REFERENCE SYSTEM: A TOOL FOR MEASURING MANAGERIAL PERFORMANCE

GELU I. VAC

Abstract. Within this paper we have assembled a proposal to build an Adaptive Reference System (ARS) which is used to evaluate organization's performance. ARS can adapt easily to any organizational environment change by tracking three very specific values: Standard Value ($ARS(Cr, SV)$), Expected Value ($ARS(Cr, ExV)$) and Actual Value ($ARS(Cr, AV)$). These values need to be acknowledged and fairly well-identified by the decision maker and/or well-computed inside each industry field in join with each evaluation criterion. Each such assembly of these values in the context of an evaluated criteria set grouped by a very specific Area of Interest (AoI) constitute a powerful *key indicator* we can use to track the organization's performance level by the chosen AoI which can be an actual department of the organization (Sales, Accountancy, etc.). The adaptability capacity of the ARS resides in its historical stored context of all the above *key indicators* which can be tracked in time to evaluate the *decision performance* by individual context (natural state).

1. Introduction

A traditional decision making system evaluates the probability of a certain scenario to take place. That means it is an activity that is evaluated BEFORE making decisions. Like this, the system is empowered with a quasi-complete set of parameters that help the managers to take the best suitable decision

Received by the editors: April 28, 2014.

2010 *Mathematics Subject Classification.* 62Cxx, 68Uxx, 90Bxx, 91Bxx, 68Mxx.

1998 *CR Categories and Descriptors.* 62Cxx [**Decision theory**]: 62C86 – *Decision theory and fuzziness*; 68Mxx [**Computer system organization**]: 68M20 – *Performance evaluation; queuing; scheduling*; 68Uxx [**Computing methodologies and applications**]: 68U35 – *Information systems (hypertext navigation, interfaces, decision support, etc.)*; 90Bxx [**Operations research and management science**]: 90B50 – *Management decision making, including multiple objectives*; 91Bxx [**Mathematical economics**]: 91B06 – *Decision theory*.

Key words and phrases. adaptive reference systems, performance evaluation systems, company evaluation systems, decision support systems, multi criteria decision analysis, key performance indicator, decision performance, organization performance, company performance.

based on that fixed context. But this only helps good/average companies to maintain their routinely course.

Making the leap from good/average to great (i.e. excellence in performance) needs more than that. To make such leap most of the time you need to "think out of the box". Thinking out of the box can be risky of course (which explains why risk-aversive Organizations have difficulties in implementing such measures), but yet extremely worthy if it turns out positively.[16] In order to train a mind to "think outside the box" you need to evaluate performance, identify thinking patterns, improve the worthy patterns (or implement them if missing) and apply them on a bigger scale. That means you need to evaluate the activity AFTER making decisions and use it as a context for future decisions.[27]

The hardest to replace in any Organization is a Manager. On the free market, when you do so, you naturally loose confidence of your shareholders, clients and more generically everybody critically depending on your services.[33] So, in order to avoid that, good/average companies reduce the risk of doing so by using decision support system tools, while the great companies reduce that risk with specific training of improving it's decision makers's mind set. [15]

In any organization, the evaluation process serves as an input to provide decision-makers with knowledge and evidence about performance and good practices.[19][29][32] Based on credible, objective, valid evidence-based information, evaluation can be a powerful tool that can make programs and projects.

Building an Adaptive Reference System for the evaluation of the business performance of an Organization is a mandatory operation. Adaptive because the market is in continuous change and companies need to keep up (adapt to new context) and performance because non-performant companies are naturally forced to leave the game of business.

This paper is the first from a series of papers dedicated to inducing performance throughout a continuous loop of measuring THE REALITY (that is the Organization's Actual Values against Industry and Technology specific criteria) and deciding next best steps to be taken in order to preserve the Organization's growth and culture.

The structure of paper is as follows. After this introductory section, next one introduces the mathematical model used. Next two sections are devoted to the configuration structure, described in the third section, and evaluated in the fourth one. Fifth section presents the results of evaluation, followed by conclusions and further work.

In order to ease perception, we have capitalized all nouns that represent the main actors of this paper.

2. Background (Related work)

2.1. **Definitions.**

Definition 1 (Business Culture)**.** *The Business Culture is related to behavior, ethics, etiquette and more. A business culture will encompass as Organization's values, visions, working style, beliefs and habits.* [18]

Definition 2 (Evaluation Framework)**.** *The Evaluation Framework is a plan that an evaluation will focus on, particular issues of importance. In particular, every framework is based on a set of underlying values and principles and an evaluation is defined as an activity that judges worth.* [14]

Definition 3 (Reference System)**.** *A Reference System is a system that uses coordinates to establish position or an organized structure for arranging or classifying.* [34]

Definition 4 (Referable Reference System)**.** *A Referable Reference System is a reference system that is capable of being assigned or credited to, capable of being referred, or considered in relation to something else.* [34]

Definition 5 (Adaptive System)**.** *An Adaptive System is a system suited, given or tending to adaptation; characterized by adaptation; capable of adapting.* [34]

Definition 6 (Adaptive Reference System)**.** *A reference system is said to be adaptive (i.e.* ***Adaptive Reference System****, ARS) when:*

- *Its set of criteria is opened in range (i.e. you can add/remove criteria from the set);*
- *The weights (level of importance) of each criteria to be measured is opened to subjectivity and natural impulse (i.e. you only set the scale range, like from 0 to 10, but when you think of a specific criteria you feel it is either 3 (not so important) or 8 (pretty important));*
- *The grouping of criteria by category is opened (i.e. you can freely create categories and subcategories like Industry specific, Technology specific, etc.) and you can freely switch/move criteria from one category to another;*
- *You can track generic entities downwards and upwards from macro to micro (i.e. either the Organization full scan, or a specific Department, a specific Team inside a Department, down to a specific Employee);*

2.2. **Motivation.** We need to perform periodical and continuous evaluation of performance in order to remain in the game of business. For instance we need to know if the company made any profit. For this reason we need to evaluate

Income versus Expenses. Evaluation is a process that critically examines a program. It involves collecting and analyzing information about a program's activities, characteristics, and outcomes. Its purpose is to make judgments about a program, to improve its effectiveness, and/or to inform programming decisions. [23][13][14]

Building a referable reference system requires criteria from within the specific technical context of that system (i.e. accountancy, sales, juridical, IT, etc.). Living now-days, forces any reference system to become extremely adaptable in order to remain a referable reference system.

While most researchers focus on how to evaluate a certain technology better or a certain technology artifact better or more structured [1][2][25], we are trying through this current study to implement a mathematical model which could help middle to top managers get the big picture of the entire functional structure performance. And of course we have a good motivation for that: this mechanism should address those managers who want to rely on this alternative.

In order to accomplish this goal, our study is based on two European pillars: first is the long forgotten pioneer of management who developed the concept of harmony between all actors involved in an Organization's production activity, that is the polish researcher Karol Adamiecki [3][17], who developed starting 1896 a production tracking diagram romantically called *Harmonograf* or *Harmonogram* and the second is the former Romanian interwar Liberal Minister of Economy who had a considerable activity as an economist and philosopher, Petre Ţuţea, who developed a personal reference system which we would like to adapt and apply to any actor designed to be involved in an Organization's production activity and for the sake of performance should be the subject of an Evaluation [30]. Those three universal references would be a person's position considering himself (self conscience), the group he/she belongs to (collective conscience) and the universe he/she belongs to (universal conscience) which we would like to analytically expose to the three methodological activities of modern research: **observation, experimentation and reasoning**.

2.3. **Goal.** Building the Adaptive Reference System (*ARS*) is the first step in performing reliable evaluation of an Organization's assets, culture and performance. It is the ground floor and foundation of an Organization's sustainability. Building a Reference System is mandatory in order to evaluate an Organization's activity and performance [21], but now-days the challenge is to be able to adapt your Reference System in order to follow the constantly changing context of the market. [22][24]

This paper proposes a method to build such an *ARS* based on Industry Standard Values and Self Expectations of Industry Values measured against THE REALITY (i.e. Actual Values - the output data of and after each Evaluation Session).

3. Main contribution

3.1. **The mathematical model.** The mathematical model should be as flexible enough to be applied from small structures like the Employee himself/herself, up-going to evaluating Teams, then Departments, and why not, looking from a market's perspective, an executive manager should be able to position his own Company among the competitors on a free market. In fact, we can already rely on a business performance model built by Jim Collins [8] which can help companies make the leap from the state of good/average to great/excellent. Also Karol Adamiecki, according to Edward Marsh's study has proven that implementing harmonical means inside a collaborative system improves productivity up to 400% [31].

We have chosen three performance indicators such as: *Standard Value* (what do others do in a similar context - and we mean here the average of competitors), *Expected Value* (how would we like to be perceived in such a context) and *Actual Value* (the reality since last evaluation as from inside the context) consciously applied to specific target should provide a relevant enough visual sight of the current behavior of the chosen target. It would not be a lie when stating that those indicators need pure talent carefully mixed with proper education and experience which is the core of a successful decision making manager. And starting from this premiss we will furtherly define each future Alternative in the Decision Matrix as depending also of the evaluation results which is there to define the additional context which has followed the implementation of a Decision Alternative.

Definition 7 (Standard Value)**.** *The Standard Value (SV) represents a numerical value explicited as from commonly available market markers of each Evaluation Criterion.*

Definition 8 (Expected Value)**.** *The Expected Value (ExV) represents a numerical value explicited as the Organizations forecasted marker of an Evaluation Criterion;*

Definition 9 (Actual Value)**.** *The Actual Value (AV) represents a numerical value computed during an internal evaluation process applied over an Evaluation Criterion;*

Definition 10 (Evaluation matrix)**.** *The* ***Evaluation Matrix*** *E (see Table 1), has the following elements:*

- *the ES rows represent consecutive Evaluation Sessions;*
- *the C columns represent Criteria that which the Organization has been evaluated;*
- *each column gathers three split sub-columns each corresponding to the three performance indicators;*
- *the Evaluation Matrix's elements represent values measured against each performance indicator of each Evaluation Criteria, as follows:*
  - *k is the market marker value of an Evaluation Criteria,*
  - *x is the expected rate value of an Evaluation Criteria and*
  - *r is the result value of an evaluated Evaluation Criteria;*

TABLE 1. The Evaluation Matrix

Evaluation Sessions	C_1			C_2			...	C_n		
	SV_1	ExV_1	AV_1	SV_2	ExV_2	AV_2	...	SV_n	ExV_n	AV_n
ES_1	k_{11}	x_{11}	r_{11}	k_{12}	x_{12}	r_{12}	...	k_{1n}	x_{1n}	r_{1n}
ES_2	k_{21}	x_{21}	r_{21}	k_{22}	x_{22}	r_{22}	...	k_{2n}	x_{2n}	r_{2n}
...	...	...	...	...	...	...	...	...	...	...
ES_m	k_{m1}	x_{m1}	r_{m1}	k_{m2}	x_{m2}	r_{m2}	...	k_{mn}	x_{mn}	r_{mn}

The classical model of a multi-criteria decision model expressed in the Table 2 [28][20], gets inreached by the Evaluation Matrix E (as is Table 1) which constitutes the continuous context that states grounds for materializing a decision alternatives tree and becomes as in Table 3.

$$D = \{A, S, R, P\} \xrightarrow{transforms\ to} D_x = \{A, S, R, P, \mathbf{E}\}$$

where diving in details shows us:

$$S = f(E) \xrightarrow{triggers} P = f(E) \xrightarrow{explains} A = f(E) \xrightarrow{as\ in} A = f(S, P)$$

Taking into account all of the above, the classical Decision matrix representation (Table 2) receives the Extended Context computed by the Evaluation Matrix (Table 3):

where:

- the Decision Alternatives A are taken in the historical context of the previous Evaluation Sessions: $A = \{A_1, A_2, \ldots, A_m\}$ - the context is supposed to be improved since you can track Decision Alternatives taken in a similar context pattern;

TABLE 2. The Decision Matrix

Decision Alternatives	*NaturalStates*			
	p_1	p_2	...	p_n
	S_1	S_2	...	S_n
A_1	r_{11}	r_{12}	...	r_{1n}
A_2	r_{21}	r_{22}	...	r_{2n}
...	...	...	...	...
A_m	r_{m1}	r_{m2}	...	r_{mn}

TABLE 3. The Decision Matrix in the context of Evaluation Session T (which constitutes the **Extended Context**)

EvaluationSession$_T$	*DecisionAlternatives*	*NaturalStates*			
		p_1	p_2	...	p_n
		S_1	S_2	...	S_n
	A_1	r_{11}	r_{12}	...	r_{1n}
	A_2	r_{21}	r_{22}	...	r_{2n}
	...	...	...	...	...
	A_m	r_{m1}	r_{m2}	...	r_{mn}

- the natural states S are situations the decision maker evaluates when building a decision alternative: $S = \{S_1, S_2, \ldots, S_n\}$;
- the results R are the consequences of each Decision Alternative in the context of a natural state, explained as quantitative measurements: $R = \{r_{ij}, 1 \leq i \leq m, 1 \leq j \leq n\}$, numbers representing the NET consequence, either a gain (if positive, $r_{ij} > 0$) or loss (if negative, $r_{ij} < 0$);
- the probabilities P are associated to the natural states S and state the probability of the corresponding natural state to take place: $P = \{p_1, p_2, \ldots, p_n\}$;

Still, disregarding *Software Development* which is a pretty mathematical field, over-numbered and over-computed [5] or *Manufacturing/Production* field which is maybe even the straighter one (you commonly have X volume of matter which goes by recipe R into Y volume of lose and Z volume of output artifacts, while X = Y + Z to check the volumes efficiency) we can compute the **Standard Value** of a given field by relying on that specific field's **evaluation criteria** and is a quest of assembling data by the subjectivity of the deciding person (manager). The 99% accurate such standards, can only be achieved on mature industries. [6][7]

Example:

(1) *Software Development:* Everybody knows there is no such thing as a bug-free software product, but rather a stable version of that software which means it has reached an acceptable balance of compromise. So on a scale of 0 to 10, such Standard Value will never be 10.
(2) *Auto Industry:* To compute the Standard Value to help you evaluate the market sales of class A vehicles, you can rely on public studies developed by various institutes of statistics and it will be different for each period of evaluation.

The **Expected Value** in this case is indeed the toughest because the deciding manager should have that special sense and should be truthful enough to envision the entity's potential in its natural context. Setting the right expectations should bring back nothing but positive results in terms of building a healthy system based on natural grounds (meaning not artificial nor artificially inseminated).

Of course units should not be scrambled so values should be measured against their siblings as well as computed to each other. This means all of these indicators should be signaled in numbers. The bigger the number which indicates the scale range, the bigger the granularity, the more accurate the evaluation.

Example:

(1) For the *Software Department* we will use a scale from 0 to 10 as in 0 for Completely Unsatisfactory, up to 10 for Extremely Satisfactory. And for the Meeting Deadline criteria fellow colleagues from the shared market agree the Standard Value is 9, out of which we can have an Expectation Value of 8.
(2) For the *Sales Department* we will use the same scale of 0 to 10, equally quoted, but, as it is field of public interest (transparency towards shareholders and fiscal authorities), Jim Collins shows us results of real studies where Standard Value is rather 3 or 4, while Expected Value is 7 and Actual Value is 10 [8].

Another question is how many criteria should we evaluate? [4] For this matter, we always need to call for the Agile perspective and that means two things: use just enough criteria to give you the right perspective and always be ready to find new measurable criteria to evaluate inside each technology depending of the goal of the measurement. One such goal should be to smother implement change within a company, just as suggested by the Schneider Culture Model [26]. Here are few examples without going to deep in search of overwhelming the system:

(1) For the Software Development Department:
    (a) Meet deadlines (estimated versus delivered);

(b) Bugs delivery (time for development versus time for fixing found bugs);

(2) For the Sales Department:

(a) Sales volume (the total amount computed from all invoices issued by the agent);

(b) Consumption (the effort used to produce the sales: phone calls, gas, etc.);

As you can see, and it is not by fortune, we have only chosen criteria which we can take numbered values from satellite software tools; that is the ERP software (which any today-company is using) for the Sales Department and Project Plan software (Microsoft Project, Version One, etc) and Bugs Management software respectively (Team Foundation Server, Bugzilla, Mantis, etc).

The basic idea is to be able to measure while producing without extra-effort and be able to measure in any given circumstances so that we can become Agile enough in taking action, reducing the risks of miss-delivery or not-delivery. In other words, the true challenge for a manager is to manifest his/her agility twice: first when choosing the right set of criteria and second when taking action according to the output results.

3.2. **The System configuration structure.** This current paper is by far not aiming to be a simple theoretical exposure, but the theoretical foundation of the actual software tool which gathers data and offers valuable output, just as expected. The goal in this case is to build a structure light enough but useful enough. With all the context analysis have taken place (that is the Standard Values and Expected Values indicators set and made peace with), we only need to configure the system so that we can use its potentially powerful reporting component afterward. Further, we use bold capitalized text to refer to the tool's entities. I started configuring the **Company** and made one step forward in flexibility allowing multiple companies (i.e. group of) to be set and like this to be the subject of evaluation for a manager implicated in such a structure on the horizontal axis of an economy exercise.

With the idea of not mixing measure units and stick to the apples basket in mind, we dive deeper and set apples apart by maturity/natural proficiency. So, after configuring all the **Departments** of a Company, we need to set, for the sake of flexibility, the biggest stage of career level for each department and for the sake of esthetic reports we will configure each **Career Level** by *Name* and unique, order-ascending *Value*.

Configuring the **Employees** will require assigning him/her to a Department and decide his/her **Employee Career Level**. We will start configuring the **Evaluation Criteria Set for each Department** and for each criteria we should map well established *Standard Value* and *Expected Value* per each

Career Level. We mean everybody knows that a Senior commonly delivers like 9 out of 10 and a Junior like 3 out of 10 so why push the Junior directly against a Senior and dis-harmonize things, when maturity comes in stages and in time and the natural way of things is to compare children with children, not with adults. For this reason, we will configure the **Evaluation Values** (Actual Values) for the unique pair *Career Level and Evaluation Criteria*.

As performing the evaluation is the key to the entire trial, we are recommending to set apart the Evaluation Criteria by Type. Like this, we will be able to track individually the **Evaluation Criteria per Type** as in *"Technology Specific"*, or *"Company Specific"* (and one could approach more specific granularity) and out of all, we should be able to track any such **Evaluation Criteria per Employee** to follow his/her career development and measure stimuli in between evaluations.

3.3. **The Evaluation configuration structure.** The software tool we have previously mentioned offers statistical output as both text and graphical reports just as you can see below. For better results, the evaluation should be organized in consecutive sessions set for a well-defined period of time. The

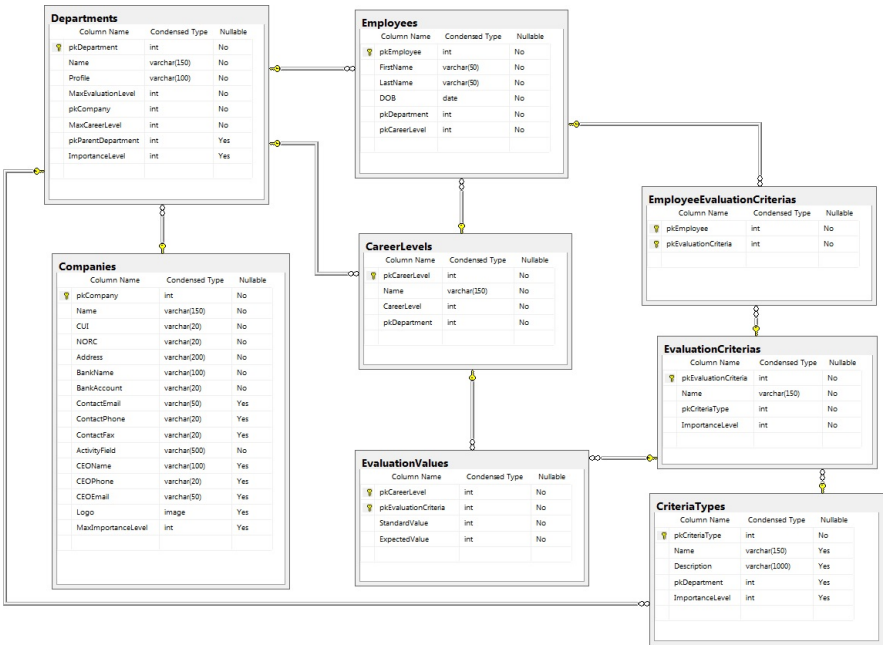


FIGURE 1. The minimal Data Base schema to configure an Adaptive Reference System

time periods should cover calendar periods not shorter than a trimester because evaluating a complex structure as this proposal is significantly time consuming (except the pivot values that can be extracted from other utility software products) AND it takes time to collect all effects of a decision.

Any **Evaluation Session** should be configured by a period of time (*Start Date* and *End Date*) and should have a friendly name, again for the sake of esthetics when creating a report and it should regard a certain Department. Like this we can track progression of either or all Employee, Team, Department and Company and supervise change.

Each Evaluation Session should keep track of the **Evaluation Feedback** so that it can be traced historically and base future decisions upon. Brief information like actual or potential Motivation which could have influenced the Employee's Evaluation Values is more than welcome to be filled-in to help soft-argumentation of a future decision.

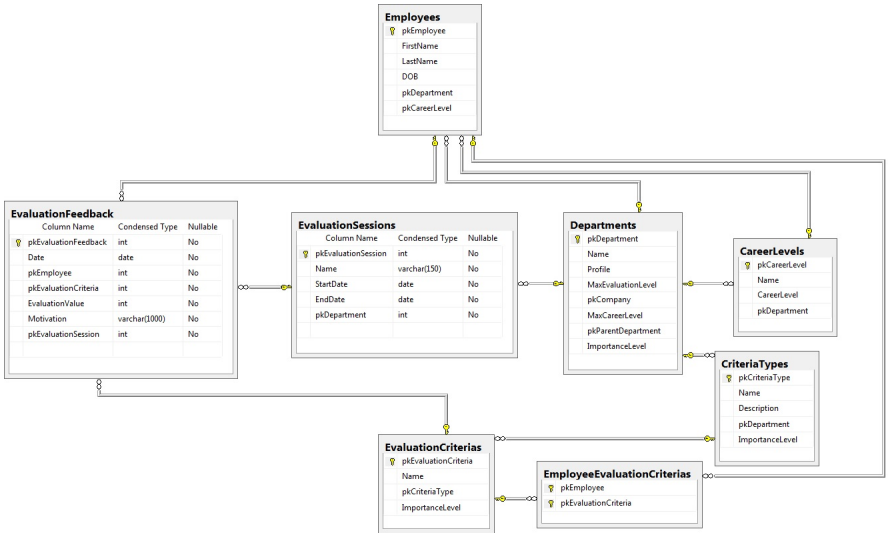


FIGURE 2. The minimal Data Base schema to configure an Evaluation Session

3.4. **Experimental evaluation results.** The compiled results will be used to develop reports and charts, significant not only for the manager, but for the Company, Department, Team and the individual Employee. The main goal of such statistical reports should not be by far the formality of personnel evaluation, but helping a manager to have a better perspective about the subordinated Entity's potential (Company, Department or Team) which should

reflect in a strategy to better position the Entity and further up the Company on a successful trajectory in terms of market.

Figures 3 and 4 contain the results of evaluating the hypothetical employee called Alan Poe hired by the hypothetical software producing company called Some Company Ltd, trying to surprise the entire contextual aspects. Evaluated Values can oftenly accede either Standard or Expected Values which can lead the decision maker to the conclusion of promoting the Evaluated Employee or motivating him/her to keep acceding, maintain the "status quo" and be able to proficiently convert this trend into material which can feed business growth. It can for sure happen that Evaluated Values are constantly bellow Standard and Expected Values which can lead the decision maker to the conclusion of either apply strategy (or different strategy) to stimulate the Employee get better results or inframe the Employee to a lower stage of Career Level where he/she could perform more accordingly.

Evaluation session		
Session name	1st trimester, 180 degree	
Evaluation period	01-Jan-13	31-Mar-13
Employee Name	Alan Poe	
Evaluation date	20-Apr-13	
Department	Software Development	evaluation scale: 10
Career Level	Senior	level 4 out of 5

COMPANY
SLOGAN / TAG LINE IF REQUIRED

Some Company Ltd
CIF: 12345678
Drive Avenue, nr. 99
The Bank account

Evaluation Criteria and results

Type	Name	Standard Value	Expected Value	Evaluation Value
Department specific (Technology specific)	Meet deadline	10	8	7
	Analysis Accuracy	7	8	9
	Time Management	7	7	9
	Coding Standards	9	8	10
	Architecturing	9	8	7
	Bugs free	9	8	8
	Coaching	6	6	8
Company specific	Apply company standards	8	10	7
	Resources management	9	9	9
	Is platform independent	5	6	3
	Estimates correctly	10	10	7
	Solves bugs fast	9	10	9
Employee specific	Communication skills	9	8	5
	Social skills	8	8	4
	Adaptability	9	10	6
	Self empowered	6	7	5

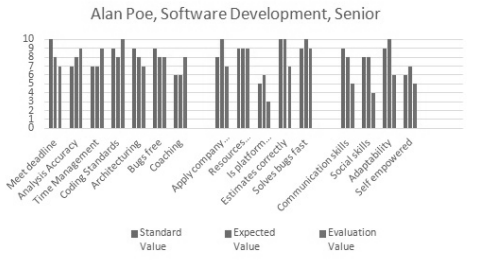


FIGURE 3. Evaluation Session results per one Employee

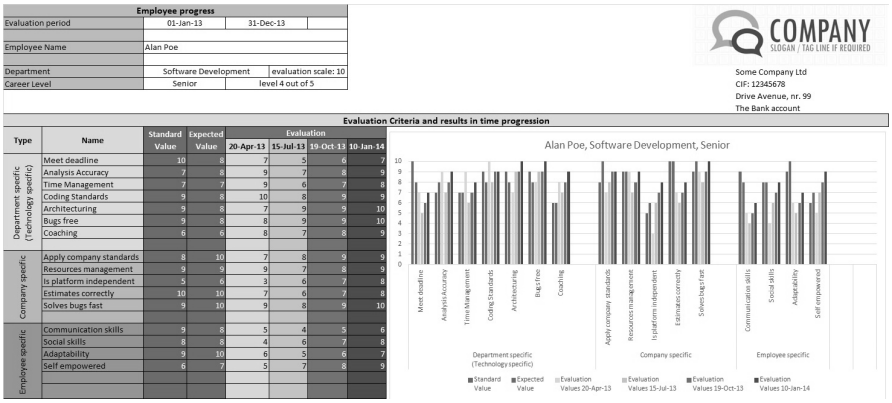
Normally, nobody makes business plans for 3 months and nobody invests in people just for fun. Improving people's professional and technical expertise brings back proficiency in process and deliverable and from there starts a chain of positive reactions which is the juice to be stimulated.

Decision making in management is a game of algorithmic if-then-else-end cases. Visual representations such as this one should help the decision maker evaluate the decisions he/she made since the previous Evaluation Session and eventually decide the next moves.

For the honest purpose of improving performance, the same Set of Evaluation Criteria should be applied periodically to the same subjects and other type of representations should be built to evaluate progress. The challenge is to keep changing and adjusting the decisions until you can identify a pattern that can inflate an ascending trend in each of an Employee's performance. [9][10][11]

Also, it will be challenge to influence an ascending trend of the same Employee on all of the evaluated Criteria. Different people have different strong areas and weak areas, so it is natural they perform different when evaluated by different criteria, BUT as long as the values of their weak areas do not affect the entity's behavior by being too weak by the Company's Expected Value.

You can see bellow a cumulative representation of the same set of Evaluation Criteria being performed with different occasions. In this case, trimestrial Evaluation Sessions which have been performed more or less at the middle of the first month after each calendar trimester end.



Employee progress			
Evaluation period	01-Jan-13	31-Dec-13	
Employee Name	Alan Poe		
Department	Software Development	evaluation scale: 10	
Career Level	Senior	level 4 out of 5	

COMPANY
SLOGAN / TAG LINE IF REQUIRED

Some Company Ltd
CIF: 12345678
Drive Avenue, nr. 99
The Bank account

Evaluation Criteria and results in time progression

Type	Name	Standard Value	Expected Value	Evaluation 20-Apr-13	15-Jul-13	19-Oct-13	10-Jan-14
Department specific (Technology specific)	Meet deadline	10	8	7	5	6	7
	Analysis Accuracy	7	8	9	7	8	9
	Time Management	7	7	9	6	7	8
	Coding Standards	9	8	10	8	9	9
	Architecturing	9	8	7	9	9	10
	Bugs free	9	8	8	9	9	10
	Coaching	6	6	8	7	8	9
Company specific	Apply company standards	8	10	7	8	9	9
	Resources management	9	9	9	7	8	9
	Is platform independent	5	6	3	6	7	8
	Estimates correctly	10	10	7	6	7	8
	Solves bugs fast	9	10	9	8	9	10
Employee specific	Communication skills	9	8	5	4	5	6
	Social skills	8	8	4	6	7	8
	Adaptability	9	10	6	5	6	7
	Self empowered	6	7	5	7	8	9

FIGURE 4. Employee retrospective over multiple Evaluation Sessions

4. Conclusions and Future Work

Defining the above *Adaptive Reference System* requires a special kind of effort and skills. The pain-point in succeeding to do it relies in the native potential of the decision maker to do the following:

(1) identifying the criteria set that meet two conditions each: it is relevant for the industry (the parent set) and it is relevant for My Organization (the subset); *and*
(2) computing, for each criterion, the values (Standard, Expected and Actual) and use each resulting key indicator as specified in this paper and by following the specific scenario for each respectively.

The benefits of using an *ARS* is to reach an accurate acknowledgment of reality and expectations and use it as solid grounds for future decisions. We have developed the tool in order to raise the accuracy of a decision in such a complex environment, where the amount of data is so big that it makes it virtually impossible to use without computed assistance. [12][13][14] The resulted evaluation should influence managerial decisions in present and future planning, strategies and policies by providing targeted recommendations to decision makers. All evaluation users should participate actively in the entire evaluation process to ensure that recommendations are practical, relevant and realistic.
Building a *Reference System* is widely accepted to be mandatory in order to evaluate an organization's activity and performance, but now-days the challenge is to be able to adapt such a *Reference System* in order to follow the constantly changing context of the market.
Building this proposal of an *Adaptive Reference System* is the first step in performing reliable evaluation of an organization's assets, culture and performance. It is the ground floor and foundation of an organization's long term sustainability.

The current paper is the baseline of argumentation for all of our future papers which are intended to give a complete survey over the everyday challenges of any organization's activity to reflect its behavioral status:

(1) Define the *Organization's Culture* using the above Key Indicators to actively and persistently evaluate it;
(2) Define mathematical models to assist an organizational to perform the *Culture leap from Actual to Expected values and goals*; and
(3) Define methodological and mathematical models to assist an organization through a *Change Perturbation in the context of the Expected Culture*.

Future notions we would also like to track and give a description to are the *Agility Indicators* of the decision makers who either build the organization's culture, build the organization's evaluation criteria set or build the motivation, means and context to respect the organization's expected culture.

References

[1] Abraham, Ajith; Vasant, Pandian; and Bhattacharya, Arijit; *Neuro-Fuzzy Approximation of Multi-Criteria Decision-Making QFD Methodology*, Springer Science + Business Media, LLC, 2008

[2] Adam, Frédéric; University College Cork, Ireland, Humphreys, Patrick; London School of Economics and Political Science, UK, *Encyclopedia of Decision Making and Decision Support Technologies*, Information Science Reference, 2008

[3] Adamiecki, Karol, *Harmonograf*, Przegld Organizacji, 1931.

[4] Baeza-Yates, Ricardo; Ribeiro-Neto, Berthier; *Modern Information Retrieval*, 1999

[5] Baggelaar, Hidde; *Evaluating Programmer Performance - visualizing the impact of programmers on project goals*, Master Thesis, Master Software Engineering, University of Amsterdam, August 2008

[6] Barthélémy, Sylvain; Filippi, Jean-Baptiste; *A Typology of Very Small Companies Using Self-Organizing Maps*, 2003

[7] Berkem, Birol, *From The Business Motivation Model (BMM) To Service Oriented Architecture (SOA)*, Journal of Object Technology, vol. 7, no. 8, November-December, pp. 57-70

[8] Collins, Jim, *Good to Great:Why some companies make the leap and others don't*, 2001.

[9] Gediga, Günther; Institut für Evaluation und Marktanalysen, Hamborg, Kai-Christoph; Universität Osnabrück, Fachbereich Psychologie und Gesundheitswissenschaften, Arbeits- und Organisationspsychologie, Düntsch, Ivo; School of Information and Software Engineering, University of Ulster, *Evaluation of Software Systems*

[10] Herlocker, Jonathan L.; Konstan, Joseph A.; Terveen, Loren G.; Riedl, John T.; *Evaluating collaborative filtering recommender systems*, ACM Transactions on Information Systems, 2004

[11] Hillston, Jane, *A Compositional Approach to Performance Modelling*, 1996

[12] Jackson, Mike; Crouch, Steve; and Baxter, Rob; *Software Evaluation: Criteria-based Assessment*, Software Sustainability Institute, November 2011

[13] Liebowitz, Jay, *Strategic Intelligence - Business Intelligence, Competitive Intelligence, and Knowledge Management*, Auerbach Publications, 2006

[14] Kahan, Barbara, *Evaluation Frameworks*, Kael Consulting, March 2008, prepared for the Saskatchewan Ministry of Education

[15] Karelaia, Natalia, HEC (Ecole des Hautes Etudes Commerciales), Université de Lausanne, *Thirst for confirmation in multi-attribute choice: Does search for consistency impair decision performance?*, Organizational Behavior and Human Decision Processes, 2006

[16] Magoc, Tanja; Center for Bioinformatics and Computational Biology University of Maryland, Ceberio, Martine; Department of Computer Science, University of Texas, Modave, Francois; Biomedical Sciences Dept., Texas Tech University, Health Sciences Center, *Using Preference Constraints to Solve Multi-Criteria Decision Making Problems*, Reliable Computing 15, 2011

[17] Marsh, Edward R., *The Harmonogram of Karol Adamiecki*, The Academy of Management Journal, Vol. 18, No. 2 (Jun., 1975), pp. 358-364, Published by: Academy of Management.

[18] Martin, Jeanette S., Chaney, Lillian H., *Global Business Etiquette: A Guide to International Communication and Customs*, http://businessculture.org/business-culture/

[19] Mehregan, Mohammad Reza; Razavi, Seyed Mostafa; Anvari, Mohammad Reza Akhavan; *Identification and Evaluation of Strategic Decisions in Gas Industry Using DEMATEL Method*, Iranian Journal of Management Studies (IJMS), Vol.5, No.2, July 2012

[20] Gokhale, Mihir, *Use of Analytical Hierarchy Process in University Strategy Planning*, University of Missouri-Rolla, Master of Science in Engineering Management, 2007

[21] Mora, Manuel; Autonomous University of Aguascalientes, Mexico, Forgionne, Guisseppi A.; University of Maryland, Baltimore County, USA, Gupta, Jatinder N. D.; University of Alabama in Huntsville, USA, *Decision Making Support Systems: Achievements, Trends and Challenges for the New Decade*, p. 86-100 Idea Group Publishing, 2002

[22] Mosavi, A., University of Debrecen, Hungary, *A Multicriteria Decision Making Environment for Engineering Design and Production Decision-Making*, International Journal of Computer Applications (0975 8887), Volume 69 No.1, May 2013

[23] Patton, M.Q., *Qualitative Research Evaluation Methods*, Thousand Oaks, CA: Sage Publishers, 1987

[24] Sahota, Michael, *An Agile Adoption and Transformation Survival Guide: Working With Organizational Culture*, 2012

[25] El-Santawy, Mohamed F.; and Ahmed, A. N.; Cairo University, Egypt, *CV-VIKOR: A New Approach for Allocating Weights in Multi-Criteria Decision Making Problems*, Life Science Journal, 2012

[26] William E. Schneider, *The Reengeneering Alternative: A plan for making your current culture work*, 2000

[27] Shamoun, Sanny, Department of Psychology, Stockholm University, *Post-decision process: Consolidation and value conflicts in decision making*, Doctoral Dissertation, 2004

[28] Tague, Nancy R., *The Quality Toolbox*, Second Edition, pages 219223 ASQ Quality Press, 2004

[29] Taillandier, Patrick; IRIT - Université Toulouse, Stinckwich, Serge; IRD/IFI/Vietnam National University, *Using the PROMETHEE Multi-Criteria Decision Making Method to Define New Exploration Strategies for Rescue Robots*, IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Kyoto: Japan, 2011, version 2 - February 2013

[30] Ţuţea, Petre, *TVR Interviews of Petre Ţuţea by Vartan Arachelian*, TVR, 1990.

[31] Urwick, Lyndall; Murphy, Mary E.; *The Golden Book of Management: An Historical Record of Seventy Pioneers*, The Accounting Review, Vol. 31, No. 3 (Jul., 1956), pp. 546-547, Published by: American Accounting Association.

[32] Velasquez, Mark and Hester, Patrick T., *An Analysis of Multi-Criteria Decision Making Methods*, International Journal of Operations Research Vol. 10, No. 2, 56-66, 2013

[33] WanAdnan, WanAdilah; MdNoor, NorLaila; Arifin, Rasimah; and NikDaud, NikGhazali, *An Experimental Evaluation of Information Visualization Techniques and Decision Style on Decision Performance*, PACIS 2007 Proceedings, Paper 62, 2007

[34] Webster Dictionary, http://www.webster-dictionary.org/

Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: geluvac@yahoo.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

DEVELOPMENT GUIDELINES FOR OPTIMIZING THE ENERGY CONSUMPTION OF MOBILE APPLICATIONS

DIANA C. ZOICAŞ

Abstract. The market of mobile devices and the power of mobile computation has increased significantly over the last years. Although the technology has evolved a lot the main issue of mobile devices is that they are and will remain severely limited by their battery life. The need to preserve this critical resource has driven mobile devices operating systems to take into consideration the power management and has driven the developers of mobile applications to optimize the energy consumption of the applications.The two main fields of research in this area are finding solutions to estimate the energy consumption of an application and finding ways to determine applications and bugs that lead to energy consumption and unexpected battery drain.

In this paper we show how we use development guidelines for mobile applications in order to determine the pieces of code that could generate a bug and could lead to an abnormal battery drain. We analyze the impact generated by the wrong usage or the lack of usage of certain development guidelines on the energy consumption. We show how the development guidelines and the best practices can be used to ensure that a mobile application is more efficient, has a better performance and consumes less energy.

1. Introduction

The market of mobile devices has increased significantly over the last years. More and more people buy mobile devices due to their usefulness and their portability. The growth of the market has also lead to an explosion of the power of mobile computation. Despite of the increased power of mobile computation the main issue of mobile devices is that they are and will remain limited by their battery life. The increased battery drain which is also called

Received by the editors: May 1, 2014.

2010 *Mathematics Subject Classification.* 68N30, 68N19.

1998 *CR Categories and Descriptors.* D.2.4 [**Software**]: Software Engineering – *Software/Program Verification*; D.2.10 [**Software**]: Software Engineering - *Design*.

Key words and phrases. mobile application, energy, energy optimization.

energy anomaly frustrates users, creates poor press for vendors and can make mobile devices unusable [8].

The construction of the batteries of the mobile devices was done mainly taking into consideration the physical size of the battery in order for the devices to be as small and light as possible and the battery capacity was not that important. The construction of the batteries has evolved a lot from the NiCD(Nickel Cadmium) bateries that were used in the 80's and 90's and were very heavy and big to the NiMH(Nickel Metal Hybride) batteries that were used in the late 90's, the Lithium Ion batteries that are still used today and to the new Li-Poly(Lithium Poly Ion) batteries which are not yet widely used but are lighter and more energy efficient than the other ones used before. The evolution of the techonlogies used for constructing the batteries is not enough for the battery life to be better so the improvement of the battery life is intended to be done in two different ways: software and processor to be less power-hungry and the devices to be constructed in order to provide the essentials and no more than that. Intel is trying to come back in the mobile world with a power-saving technology that will be constructed on a platform that is already available for the light laptops.

The need to preserve energy has driven mobile devices operating systems concentrate a lot at the power management. The two main fields of research in this area are the implementation of tools and techniques to estimate the energy consumption of different applications and the implementation of tools and techniques to discover applications and bugs that lead to energy consumption and unexpected battery drain. A research field that is still under development is the implementation of tools that would identify the code that could be improved regarding energy consumption and provide solutions for optimizing the code.

As a first step in the process of developing such a tool in this paper we are identifying the development guidelines that need to be followed in order for the mobile applications to do not generate energy bugs. The energy bugs are defined as being an error in the system, either application, operating system, hardware, firmware or external that causes an unexpected amount of high energy consumption by the system as a whole [1]. We are analyzing some of the development guidelines that can optimize the energy consumption of a mobile application. We are focusing on four types of guides: general development guidelines, data manipulation guidelines, performance guidelines and background jobs guidelines. We are identifying the reason for which these guidelines can improve the energy consumption.

2. Background

It happens sometimes that the battery of a mobile device drains very fast. This drain is caused by energy leaks which can have two causes: hardware bugs or software bugs.

The hardware bugs can be caused by any defect of a hardware part of the mobile device. Hardware bugs are bugs that are not related to the implementation and are caused by some hardware components. The first hardware bug is related to a faulty battery. This bug is mainly solved by replacing the battery. Another hardware energy bug is related to the exterior hardware damage. The SIM card can also cause battery drain in multiple ways. An external SDCard can also trigger severe battery drain. Another generator of energy bugs is also an external hardware (phone chargers, external docks used for recharging or for audio capabilities)

The software bugs can be generated by the operating system that is installed on the mobile device, by a certain application or by a programming mistake.The first category of software bugs are the operating system (Android, iOS, Windows Mobile) bugs that are generated by an update that was done by the user or by an automatically done update. In most of the cases the solution for this type of bugs is to do a downgrade of the operating system. The second category of software bugs are the application and framework energy bugs and the most known bugs of this type are the No-Sleep Bugs [6]. The application and framework bugs are the bugs that are generated by the implementation of an application or are generated by the framework that is used when developing a mobile application. The root cause of these bugs can be anything from a simple implementation error to complicated reasons like race condition that prevent the lock release. Another known energy bug is the loop bug. In this case a part of an application enters a looping state and performs periodically unnecessary tasks. The third sub-category is the immortality bug. The behavior of the application is the following: it is killed and it restarts. The third category of energy bug are the energy bugs triggered by External Conditions [1]. One of the external conditions that influence the battery drain is the Wireless Signal Strength [1], Wireless Handovers [1]. Besides the above energy bugs there are also unknown bugs that were reported but for which the root cause is unknown. There are a few tools that help the user to narrow down to suspicious application that generates the energy bugs.

There are some software bugs that are widely known for different operating system of the mobile devices. For Android there are two main types of energy bugs: no-sleep bugs and loop bugs. A no sleep bug occurs when the CPU is waken up by an application but it is never put back to sleep therefore excesively consuming the energy without providing any functionality [5]. The

loop bugs are ocurring when a thread is waiting for a certain event in order to continue and a variable is used when testing the condition. The thread will continuously poll the variable until it is changed and therefore it consumes CPU without doing any work for the user [5]. The power models of Android and Windows are similar so also the energy bugs from Windows are similar to the ones from Android. In the iOS system an application can be only in one of the four states that are definded and the handling of the states is fully done by the developers [7] so the energy bugs can be caused by the prolonging of the transition between two states and can be caused only by the developer.

There are more types of *No-Sleep Bug* and three of these types are *No-Sleep Code Paths*, *No-Sleep Race Condition* and *No-sleep dilation* [2].

No-Sleep Code Path (Figure 1): The root cause of most of the bugs is the existence of a code path in the application that wakes up a component but does not put the component back to sleep. The first cause is that the programmer forgot to do the release through the code or he has put it on a conditional path but not on all the paths. Another cause is that the programmer did put the release code but the code took an unanticipated code path during execution and the release was not executed.

```
try{
      res.acquire();//CPU should not sleep
      file_load();//a method that thorws exception
      res.release();//CPU can sleep
}
catch(Exception e) {
      e.printStackTrace();//log the error
}
finally{
} //end of try block
```

FIGURE 1. No-Sleep bugs: code paths

No-Sleep Race Condition: These bugs were caused by race conditions in multi-threaded applications. The power management of a particular component was carried out by different threads in the application (one thread switches the component on and later another thread should switch it off) and the sequence of execution was a different one that expected.

```
public void First_Thread(){
      killFlag = false; //kill flag unset
      res.acquire(); //CPU should not sleep
      start(inner_thread); //Start inner_thread
      //....Execute logic
      killFlag = true; //kill flag set
      stop(inner_thread); //Signal inner_thread
      wl.release(); //CPU can go to sleep
} //End First_Thread();

public void Inner_Thread(){
      while(true){
            if(killFlag) break; //Break if flag true
            file_load(); //might throw exception
            res.release(); //release before sleep
            sleep(180000);//Sleep for 3 minutes
            res.acquire(); //CPU should not sleep
      } //End while loop;
} //End Inner_Thread()
```

FIGURE 2. No-Sleep bugs: race condition

No-Sleep Dilation: the component woken up by the application is ultimately put to sleep by the application but only after a substantially longer period of time than expected or necessary.

In order to develop an application that is energy efficient it is not enough to have an application that does not generate energy bugs. We have to pay attention to the implementation that could be enhanced from the energy point of view. From the previous papers we have seen that most of the research was concentrated on implementing tools that analyze the code for energy leaks and that find energy bugs but none of these tools offer also guidelines for the developer on how to fix the issues that were discovered.

3. Main contribution

Most of the papers related to optimization of the mobile application in order to reduce the energy consumptions are describing techniques and tools that help the developers to determine the source code that generates energy leaks. These tools (ADEL, Carat, eDoctor) help developers to discover the root cause of the error but they do not offer guidelines on how the developers should improve the code. Nowadays the developers can find solutions for optimizing the energy consumption by searching on the web for an optimal solution, by consulting the specialized forums or by using the previous experience of the developer. From our point of view it is more important to implement the mobile application in such a manner that the energy leaks are reduced to minimum. It is cheaper to write correct and efficient code from the implementation phase than to re-write the code for optimizing it to reduce the energy consumption. This is the reason for which we consider that it is important to determine development guidelines that can be used by the developers in order to write correct and efficient mobile applications.Taking this in consideration, a research field that is still under development is the implementation of tools that would identify the code that could be improved regarding energy consumption and provide solutions for optimizing the code.

As a first step in the process of developing such a tool in this paper we identify the development guidelines that need to be followed in order for the mobile applications to do not generate energy bugs. We are analyzing some of the development guidelines that can optimize the energy consumption of a mobile application. We first try to identify from development guidelines provided by the different mobile devices operating systems vendors the fields that could be improved in order for the energy consumption to be reduced to a minimum. Due to the fact that the Android operating system is the most used one and the API system is presented more clear, we have tried to focus our research in this area but keeping also in mind the other operating systems for mobile devices. The identifcation of the development guidelines to be used for optimizing the energy consuption we have only taken into consideration the API that is exposed by the operating systems.We have identified different types

of development guidelines that can influence the energy consumption of mobile applications: general development guidelines, data manipulation guidelines, performance guidelines and background jobs guidelines. In the future papers we wil start the implementation of the tool with the automatic identification of the development guideline that we discovered in this paper. This identification will be done for each of the discovered development guidelines.

General development guidelines. The first guidelines that should be taken into consideration when developing a mobile application are the ones for avoiding the coding errors. First of all it is important that the code of the application does not produce energy bugs. For each of the No-Sleep bugs there is a guideline to be used in order to avoid the respective bug. As mentioned in the above sections, the most important energy bugs are the No-Sleep bugs.

The root cause of the No-Sleep Code Path energy bug, as it can be seen in Figure 1, is that the resources that are being used are not released on all the paths so the CPU cannot go back to sleep and it continues to consume energy. This bug can be avoided by verifying that the code for releasing the resources that are being used is present in all of the paths of the method that acquires the respective resource. In the case of Figure 1 the solution would be to add the code for releasing the resources in the *finally* block.

The No-Sleep Race Condition energy bug can be present in all the multi-thread mobile applications. As it can be seen in Figure 2 , the problem occurs when the handling of the power management for a certain component is done in different threads. In a normal execution path of the threads the component would be acquired and released as expected but in some exceptional cases the path could be different and the resource will not be released. In the case of No- Sleep Race Condition the developer should pay special attention to the execution path for the threads [6] and make sure that the components that were acquired are released no mather the order in which the threads get executed.

Data manipulation guidelines. One of the important aspects regarding the functionality and energy consumption is related to data manipulation. The energy consumed by operations for data manipulation depends on the number of calls that are done for sending/saving data and on the size of the data that is being manipulated. In paper [3] it was determined that it is more efficient sending larger files than sending smaller files. Due to these findings, a guideline that would optimize the energy consumption would be to gather smaller data files into one bigger file and send once as much data as possible. This guideline could also be applied to the HTTP requests. It is recommended to bundle multiple small requests into one bigger request in order for the request to be more energy efficient. For verifying that more data

is sent at once and not in smaller chuncks we could check that all the calls for saving the data are gathered in only one call.

Performance guidelines. The performance of an application is directly proportional to the energy consumed by the respective application. If the application runs faster it means that the resources are released earlier and the CPU can enter the sleep state earlier. The code that has a better performance is consuming in some of the cases more resources than a code which has not such a good performance. Despite of this fact it was measured that the energy consumption of a code with a better performance consumes less energy [3]. There are a lot of development guidelines for improving the performance of an application. The first and most important guideline is to do not create unnecessary objects. Each of the instantiated objects needs a garbage collection that consumes energy when it runs so we should check in our tool that if an object is declared, it is also used in the methods in which it was declared. We should also check the global variables to be used in the code of the application. Another important development guideline, at least on the Android OS, is to rather use the fields of a class directly than by calling the getters and the setters [4]. In Android it is much more expensive to call the getters and setters than the fields directly. When testing both the setters and getters and the direct access methods it results that the energy consumption decreases by 30% when accessing directly the fields and not via the getters and setters. We will check in our tool that, when using a certain variable we are accessing it directly and not via the getters and setters.

Background jobs guidelines. One of the energy consuming tasks within an application are the background jobs and the optimization for these jobs is really important. For optimizing the energy consumption of the background jobs the most important development guideline that should be followed is to group more background jobs in order to run at the same time. In this way the resources are acquired at the same time and are released at the same time and allow the CPU to be in a sleep state for a longer period. Another development guideline that should be taken in consideration when implementing background jobs is that data for the background jobs should be acquired though an asynchronous call. In this manner the application will not wait for a response and it will let at least the screen in a sleep mode.

4. Conclusions and Further Work

The users of the mobile devices tend to reject the applications that are generating energy leaks in order to benefit as much as possible of the usability of their mobile devices. Due to this new requirement the investment in finding

solutions for optimizing the energy consumption became a very important research topic. Most of the research that is done for the optimizing the energy consumption is oriented towards techniques that try to identify where the energy is lost and which is the amount of the energy that is lost.

From our point of view the research should be oriented towards identifying the development guidelines and best practices that can lead to the development of energy-efficient applications. In this paper we have identified some of the development guidelines that help mobile application developers to write applications that do not produce energy bug. We have also identified some development guidelines that help mobile application developers to implement applications that will be more energy-efficient. For future work we will investigate more development guidelines that could lead to an improvement of energy consumption of mobile application. We will also focus more on the development guidelines for Blackberry OS, iOS or Windows Phone OS. We also would like to implement a tool to check that the implementation of a mobile application complies to the development guidelines that should be used for the energy consumption optimization. We would also like for this tool to offer solutions for improving the pieces of code that are not energy-efficient.

References

[1] Abhinav Pathak, Y. Charlie Hu, Ming Zhang, *Bootstrapping Energy Debugging on Smartphones: A First Look at Energy Bugs in Mobile Devices*, Proceedings of the 10th ACM Workshop on Hot Topics in Networks, New York, USA, 2011, Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.232.9209

[2] Abhinav Pathak, Abhilash Jindal, Y. Charlie Hu and Samuel P. Midkiff, *What is keeping my phone awake? Characterizing and Detecting No-Sleep Energy Bugs in Smartphone Apps*, Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, Lombard, IL, USA, April 2013, Available: https://www.usenix.org/system/files/conference/nsdi13/nsdi13-final198.pdf

[3] Ding Li, William G. J. Halfond, *An investigation into Energy-Saving Programming Practices for Android Smartphone App Development*, Proceedings of the 3rd International Workshop on Green and Sustainable Software (GREENS), Hyderabad, India June 2014, Available: http://www-bcf.usc.edu/ halfond/papers/li14greens.pdf

[4] Ding Li, William G. J. Halfond, *An Investigation into Energy-Saving Programming Practices for Android Smartphone App Development*, Proceedings of the 3rd International Workshop on Green and Sustainable Software (GREENS), Los Angeles, USA June 2014, Available: http://www-bcf.usc.edu/ halfond/papers/li14greens.pdf

[5] Jack Zhang, Ayemi Musa, Wei Le, *A Comparison of Energy Bugs for Smartphone Platforms*, 1st International Workshop on the Engineering of Mobile-Enabled Systems, San Francisco, USA 2013, Available: http://www.cs.iastate.edu/ weile/docs/le-mobs13-1.pdf

[6] Panagiotis Vekris, Ranjit Jhala, Sorin Lerner and Yuvraj Agarwal, *Towards Verifying Android Apps for the Absence of No-Sleep Energy Bugs*, Proceedings of 2012 Workshop

on Power-Aware Computing and systemsHotPower 2012, Hollywood, CA, Available: http://dl.acm.org/citation.cfm?id=2387872

[7] Shuai Hao, Ding Li, William G. J. Halfond, Ramesh Govinda, *Estimating Mobile Application Energy Consumption using Program Analysis*, Proceedings of the 2013 International Conference on Software Engineering, Pages 92-101, Piscataway, NJ, 2013, Available: http://www.cs.binghamton.edu/ millerti/cs680r/papers/EstimatingMobileApplication Energy.pdf

[8] Xiao Ma, Peng Huang, Xinxin Jin, Pei Wang, Soyeon Park, Dongcai Shen, Yuanyuan Zhou, Lawrence K. Saul and Geoffrey M. Voelker, *eDoctor: Automatically Diagnosing Abnormal Battery Drain Issues on Smartphone*, Proceedings of the 10th ACM/USENIX Symposium on Networked Systems Design and Implementation, Lombard, IL, April 2013, Available: http://cseweb.ucsd.edu/ voelker/pubs/edoctor-nsdi13.pdf

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania

E-mail address: diana.zoicas@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

DESCRIPTORS FUSION AND GENETIC PROGRAMMING FOR BREAST CANCER DETECTION

ŞTEFANA FRĂTEAN AND LAURA DIOŞAN

Abstract. The detection of tumors in digital images originated from mammograms can be a challenging task. In this paper we investigate a Computer Aided Diagnosis System based on a Genetic Programming classifier. The performance of the considered classifier is evaluated for five of the image descriptors used in literature and we propose a new approach, by utilizing descriptors fusion. Numerical experiments are performed on a sample of the Digital Database for Screening Mammography data set and indicate that descriptors fusion can lead to better classifying performances.

1. Introduction

Breast cancer is the second most frequent form of cancer in the world, and, by far, the most frequent form of cancer among women, with over 1.67 million new cases diagnosed in 2012 (25% of all cancer cases) [8]. Even though early, asymptomatic stages of breast cancer can be detected using a non-invasive technique, mammogram examination [16], this approach relies mostly on the expertise of the radiologist and can lead to a high number of erroneous diagnoses. According to [10], about one in 2000 women will have her lifespan increased by ten years due to mammogram examination, while other ten women will be administrated unnecessary treatment. Moreover, 200 women will suffer from significant psychological stress due to false positive results. Another reason for concern is the false negative diagnoses, as it is estimated that about 10% - 30% of the breast cancer cases are never detected using mammograms.

As a solution to this problem, a technique called double reading has been adopted [18]. More precisely, each case is analyzed independently by two radiologists, in an attempt to reduce the rate of wrong diagnoses. However, since this leads to additional costs and workload, Computer Aided Diagnosis

Received by the editors: October 23, 2014.

2010 *Mathematics Subject Classification.* 68T05, 91E45.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Induction.*

Key words and phrases. Genetic programming, Image descriptors, Descriptors fusion, Breast cancer.

Systems can assist a single radiologist in interpreting the mammogram, offering him support in establishing a diagnostic.

The purpose of this paper is to investigate a genetic programming (GP) classifier for the learning phase of such a system. The input data for the classifier will consist of image features extracted using several different descriptors, namely Statistical Moments, Gray Level Run Length (GLRL), Haralick features, Gabor filters and the Histogram of Oriented Gradients (HOG). Also, we will assess the performance obtained by combining (through fusion) image descriptors as, to the best of our knowledge, this approach has not been used before for breast cancer detection.

The paper is organized as follows: Section 2 details the breast cancer diagnosis problem and the main steps that must be performed for solving it. Section 3 describes the utilized image descriptors, Section 4 reviews the main components of the proposed learning algorithm, and Section 5 shows the numerical experiments and the obtained results. Conclusion and future work can be found in Section 6.

2. Problem of breast cancer diagnosis

The problem of breast cancer diagnosis has been intensively studied as a binary classification problem and has been solved by a supervised learning algorithm by respecting the architecture of a general classifier of two main modules: one for data pre-processing (in fact, image processing) and another one for induction of the classifier.

The input data for such a problem consists in a set of training data examples (in our case features extracted from images) each labeled correspondingly as positive or negative samples depending on which class (healthy patient or sick patient) they belong to. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. The data set is split in two parts: training data and testing data. The learning algorithm will construct the decision model by using both information (image descriptor and class) about all the images from the training set. After the classifier has been constructed, its performance will be verified by using the testing data (the classifier outputs will be compared by the real labels associated to each image from the testing set). Following this scenario, the algorithm will be able to correctly determine the class labels for unseen instances.

Different machine learning classifiers have been trained on features like GLRL [12], Statistical Moments [2], Gabor filters [19], Haralick features [21]. From the previously used classifiers, we remember Support Vector Machines, Random Forests, Logistic Model Trees, K Nearest Neighbors, and Naive Bayes.

In [15], image descriptors have been evaluated in the presence and absence of clinical data and the HOG feature descriptor has been used for the first time for describing breast lesions.

In this paper, we propose a new approach that involves the use of a GP classifier that encodes discriminant functions, and the use of the area under the Receiver Operating Characteristic (ROC) curve as a fitness measure. Furthermore, we show that for the given problem, descriptors fusion outperforms the use of individual descriptors, the system obtaining a maximum accuracy of 0.87 when using the combination of Haralick features, moments and GLRL.

3. IMAGE DESCRIPTORS

The first step in building automated diagnosis systems is the extraction of relevant characteristics from images. In order for a system to be able to classify images in different categories, first these need to be processed, resulting in numerical values that the system can interpret. Therefore, representations of visual characteristics like shape, color and texture, called image descriptors or visual descriptors, need to be extracted from the images.

3.1. **Statistical Moments.** Statistical Moments are statistical measures based on the intensity of the pixels of an image, and are computed from the gray level image histogram. In this paper we include a set of six features: mean value, standard deviation, skewness, kurtosis, the minimum and the maximum intensity value.

3.2. **Grey Level Run Length.** Grey Level Run Length (GLRL) [9] computes the occurrences of sets of correlated pixels, for a given length, direction, and for a particular gray level, and stores these values on a GLRL matrix. More precisely, being given a direction (e.g. the horizontal direction), for each allowed gray level it is checked how many sets of two consecutive pixels with the same value exist. Then the procedure is repeated for groups of three pixels, four pixels and so on. From the resulted matrix different characteristics can be computed, resulting in a set of 11 features.

3.3. **Haralick features.** The gray level co-occurrence matrix represents a technique of computing statistical measures of the distribution of pixel pairs within an image, by considering the relationships between two pixels, called the *reference* pixel and the *neighbor* pixel. Starting from the top left corner and continuing to the right down corner, each pixel becomes in turn a reference pixel. Given a separation distance, also called *offset*, occurrences of pairs of pixel with a certain gray level are counted and added to the matrix. Out of this matrix Robert M. Haralick has proposed the extraction of different characteristics that describe texture, called Haralick features [5], out of which

we mention the energy, entropy, contrast, inverse difference moment, inertia and correlation.

3.4. **Gabor filters.** Gabor Filters are linear filters that model the functions of simple cells from the mammalian visual cortex. Therefore, image processing using Gabor filters is thought to be similar to the perceptions from the human visual system [14]. More precisely, considering the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave [4]. Gabor filters are often used for edge detection, as they detect edges with a given frequency and orientation.

3.5. **Histograms of Oriented Gradients.** The principal idea behind the Histograms of Oriented Gradients (HOG) descriptor is that object appearance and shape within an image can be described by the distribution of intensity gradients. More precisely, HOG is an image descriptor used for object detection which computes the number of gradients orientation in localized portions of an image [3]. The image is divided in small spatial regions, called cells, and for each cell a 1-D histogram of gradient orientation is computed. The first step in computing the histogram is represented by the gamma and color normalization. Then the gradient magnitude is computed for each pixel, by applying certain mask filters. After that follows the computation of the gradient orientation, which is the direction of the fastest gradient change, obtaining a matrix with n x m values, where n and m represent the dimensions of the image in pixels. From this matrix the HOG is computed by counting for each cell the number of pixels for which the gradient orientation falls in the respective cell. Finally, in order to obtain invariance to illumination and shadowing, the histogram can be normalized, common options being the L1 and L2 norms [3].

4. GENETIC PROGRAMMING

After extracting relevant features from the images, the next step is building a machine learning system that will use those features as input data. For this purpose, we have decided to use a Genetic Programming (GP) algorithm [13] which, after learning from annotated examples, will be able to generalize and classify newly seen mammograms. GP algorithms were chosen because they are a flexible and powerful evolutionary technique. GP are able to both represent data and to perform computations, and can be used not only for classifying data, but also for data pre-processing and post-processing. Moreover, due to the fact that the discriminant function evolved by the algorithm

is similar to the mathematical operations and transformations used in image processing, GP algorithms are considered to be very suitable for image classification tasks [6].

We have chosen to use a tree representation of chromosomes, with nodes consisting of image features and constants, and the functions {+, -, *, /}, where +, - and * represent the usual mathematical operators and / is the safe division (given a and b, it returns a / b if $b = 0$ and 1 otherwise). The genetic operators are the usual ones: crossover is performed by switching two randomly selected sub-trees, mutation consists of changing a randomly selected sub-tree with a newly generated tree and the selection method is tournament selection. The population is initialized with the ramped half-and-half method [13].

For GP algorithms, the classical approach in computing the fitness measure in the case of a binary classification problem is choosing one or several threshold points and evaluating, for each of them, the value corresponding to the true positive and false positive rates. In the case of multiple threshold, we will obtain a ROC curve, and the fitness will be obtained by computing the area underneath [7]. However, as this method can lead to increased execution time, we have proposed a new approach that, as far as we know, has not been tried on mammograms, by using the Wilcoxon-Mann-Whitney (WMW) statistic [20] to estimate the area under the ROC curve. WMW does not require the actual building of the ROC curve and therefore is less expensive in concerns of execution time. The basic thought is that pairwise comparisons of the negative class observations and positive class observations are performed, collecting rewards when certain constraints are satisfied. The first constraint is the fact that the observations of the positive class need to be correctly labeled (e.g. bigger than zero) and they also need to be bigger than the observations of the negative class [1]. In this way, WMW is efficient not only in evaluating the accuracy of the classifier on the training data, but also in separating the instances of the two classes.

5. Numerical results

5.1. **Data Set.** In order to evaluate the performance of the considered classifier, *The Digital Database for Screening Mammography* (DDSM) [11] has been chosen due to the high number of annotated mammograms with biopsy proven diagnostic. The data set was originally build from two types of film mammograms, mediolateral oblique and craniocaudal, which were then scanned to obtain digital images in LJPEG format. In order to be able to process those images, we converted them in PNG format, using the freely available program

DDSM Software [17]. The resulted images have a size ranging from about 1,600 x 3,700 pixels to about 3,800 x 6,800 pixels.

Out of the 2620 available cases, we have chosen a sample of the mammograms from the normal volumes 1-6 and all the available images from the cancer volumes (volumes 1-15), resulting in a number of 3555 images from four different scanners. About two thirds of the images were used for training the classifier, and the rest for validating the system.

5.2. **Experimental framework.** Experiments were performed for each of the considered descriptors taken individually and for seven descriptor combinations, resulting in 12 scenarios. For each scenario 200 experiments were ran and the performance of the best solution, of the worst solution and the mean performance of all of the solutions were evaluated. The classification performance was measured by computing the accuracy, the precision and the recall and, for each measure, confidence intervals of 95% were provided.

For each experiment we considered a population of 100 individuals that evolved during 500 generations or until no changes were recorded in the population for at least 100 generations. The mutation and the crossover probabilities were the ones recommended by J. R. Koza, namely 10% and, respectively, 90% [13]. The maximum depth of each chromosome was computed by the formula from Eq. 1 [13] where *terminalsNumber* represents the number of terminals.

$$\frac{\log(terminalsNumber)}{\log(2)}, \tag{1}$$

5.3. **Individual descriptors results.** Out of the five considered descriptors, the best classification performance was recorded by statistical moments, which obtained the highest scores for maximum accuracy, maximum precision and for all of the performance measures for the mean and for the worst solution. The maximum accuracy was equal to 0.83 and the maximum recall was 0.86, both registered by the moments descriptor. The maximum precision, of 0.83, was scored by Haralick features. Figure 1 shows complete results, with confidence intervals, for accuracy, precision and recall.

5.4. **Descriptors fusion results.** Out of the five considered descriptors, we chose for the fusion the four descriptors that scored the best performances when evaluated individually: statistical Moments, Haralick features, Grey Level Run Length and Histogram of Oriented Gradients. In general, the maximum classification performances were better when using descriptors fusion: out of the seven tried fusions, five scored a maximum accuracy that was better than the maximum accuracy obtained by individual descriptors. Furthermore, one other fusion (HOG + GLRL) had a slightly better accuracy than the ones

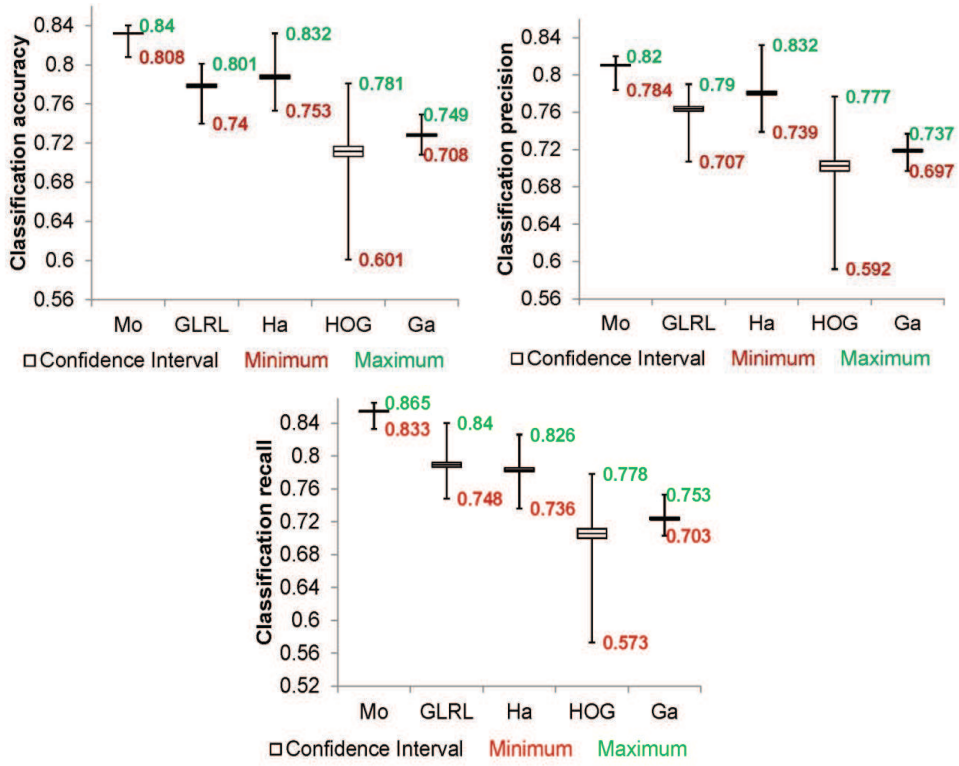


FIGURE 1. Classification results of individual descriptors. Mo: statistical moments; Ha: Haralick features; Ga: Gabor filters.

obtained by using the two descriptors individually (0.8042 for the fusion, compared with 0.8008 for GLRL and 0.7814 for HOG). The best performance of the system resulted from using the combination of Moments, GLRL and Haralick features. For this fusion, the maximum accuracy was of 0.87, the maximum recall was of 0.85 and the maximum precision was equal to 0.88. Figures 2 shows the results obtained by using descriptors fusion.

5.5. **Comparison to related work.** In order to assess the performance of the GP classifier, we analyzed the results of individual descriptors with the ones reported in the study [15], where Support Vector Machines, Random Forests, Logistic Model Trees, K Nearest Neighbors and Naive Bayes classifiers were used. However, only one set of results was presented, and the authors do not mention to which learning algorithm it belongs.

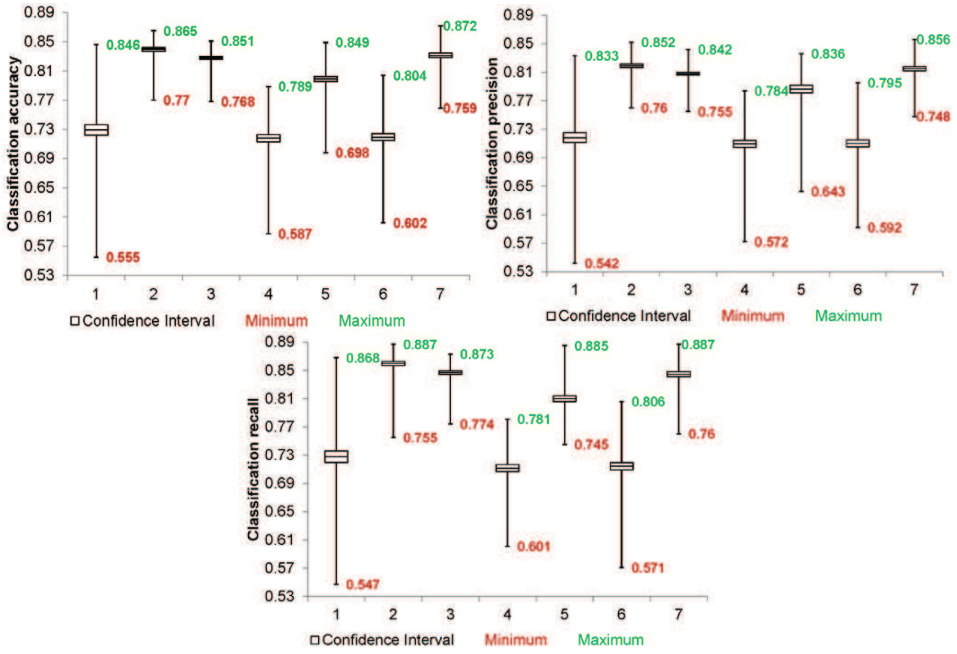


FIGURE 2. Classification results of descriptors fusion. 1: Mo + HOG; 2: Mo + Ha; 3: Mo + GLRL; 4: Ha + HOG; 5: Ha + GLRL; 6: GLRL + HOG; 7: Mo + Ha + GLRL.

For a more accurate comparison, we only considered the results obtained using an experimental framework similar to ours, where images were sampled from the DDSM data set and no clinical data was used. Table 1 compares the mean accuracies obtained by the GP classifier with the accuracies presented in [15], showing that GP performed better for all of the considered image descriptors. However, since these results were obtained using different images, in different numbers, the comparison does not have a statistical characteristic and can only serve as a guideline.

TABLE 1. Mean accuracies of the GP compared with the ones reported in [15].

	GP	[15]
Moments	0.831	0.707
GLRL	0.778	0.733
Haralick	0.787	0.718
HOG	0.711	0.707
Gabor	0.723	0.711

6. CONCLUSIONS

Breast cancer diagnosis can be a difficult process that involves high workload and relies mostly on the expertise of the radiologist. In order to reduce the chance of human error while decreasing the costs implied by double-reading, automated diagnosis systems could aid the radiologist in taking a decision and determining a diagnostic. This paper investigates a new approach to such a system, by using a Genetic Programming algorithm. Therefore we show that GP is suitable for the task of mammogram classification, the system obtaining a maximum accuracy of 0.87. Furthermore, we propose the use of descriptors fusion, and we show that in most of the cases this approach outperforms results obtained by individual descriptors.

Future work includes testing all other possible fusions of the considered descriptors in order to verify whether a less efficient descriptor could lead to an increase in the overall performance of the combination. Another possible direction is evaluating different machine learning classifiers (e.g. Support Vector Machines) on the given data sets, in order to be able to perform a statistical comparison with the performance of the GP classifier. Also, we intend to validate the obtained results on other data sets such as BCDR and MIAS.

REFERENCES

[1] U. Bhowan, M. Zhang, and M. Johnston. Improving gp with new fitness functions. In *Genetic Programming 13th European Conference Proceedings*, pages 3–6, 2010.

[2] I. Christoyianni, E. Dermatas, and G. Kokkinakis. Fast detection of masses in computer-aided mammography. *IEEE Signal Proc Mag*, 17(1):54–64, 2000.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893, 2005.

[4] J. G. Daugman. Uncertainty relation for resolution in space, spatial-frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of Optical Society of America*, 2(7):1160–1169, 1985.

[5] I. Dinstein, K. Shanmugam, and R. M. Haralick. Textural features for image classification. In *CMetImAly77*, pages 141–152, 1977.

[6] Pedro G. Espejo, Sebastian Ventura, and Francisco Herrera. A survey on the application of genetic programming to classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(2):121–144, March 2010.

[7] Tom Fawcett. ROC graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4, Hewlett Packard Laboratories, June 2 2003.

[8] J. Ferlay, I. Soerjomataram, M. Ervik, R. Dikshit, S. Eser, C. Mathers, M. Rebelo, D. M. Parkin, D. Forman, and F.Bray. Globocan 2012 v1.0, cancer incidence and mortality worldwide: Iarc cancerbase. Technical Report 11, International Agency for Research on Cancer, Lyon, France, 2013.

[9] M. M. Galloway. Texture analysis using gray level run lengths. *Computer Graphics Image Processing*, 4(2):172–179, June 1975.

[10] P. Gotzsche and M. Nielsen. Screening for breast cancer with mammography. *The Cochrane Library*, 2011.

[11] M. Heath, K. Bowyer, D. Kopans, R. Moore, and P. Kegelmeyer. The digital database for screening mammography. In *Proceedings of the Fifth International Workshop on Digital Mammography*, pages 212–218, 2001.

[12] J. K. Kim and H. W. Park. Statistical textural features for detection of microcalcifications in digitized mammograms. *IEEE Trans. Medical Imaging*, 18(3):231–238, March 1999.

[13] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[14] S. Marcelja. Mathematical description of the responses of simple cortical cells. *Journal of Optical Society of America*, 70:1297–1300, 1980.

[15] Daniel C. Moura and Miguel Ángel Guevara-López. An evaluation of image descriptors combined with clinical data for breast cancer diagnosis. *Int. J. Computer Assisted Radiology and Surgery*, 8(4):561–574, 2013.

[16] H. D. Nelson, K. Tyne, A. Naik, C. Bougatsos, B. K. Chan, and L. Humphrey. Screening for breast cancer: systematic evidence review update for the us preventive services task force. *Ann Intern Med*, 151(10):727, 2009.

[17] D. C. Rose. Ddsm software. ttp://microserf.org.uk/academic/Software.html. Accessed: 2014-06-03.

[18] L. Tabar, B. Vitak, T. Chen, A. Yen, A. Cohen, T. Tot, S. Chiu, S. Chen, J. Fann, J. Rosell, H. Fohlin, R. Smith, and S. Duffy. Swedish two-county trial: impact of mammographic screening on breast cancer mortality during 3 decades. *Radiology*, 260(3):658–663, 2011.

[19] Defeng Wang, Lin Shi, and Pheng-Ann Heng. Automatic detection of breast cancers in mammograms using structured support vector machines. *Neurocomputing*, 72(13-15):3296–3302, 2009.

[20] Lian Yan, Robert H. Dodier, M. Mozer, and Richard H. Wolniewicz. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. 2003.

[21] S. Yu and L. Guan. A CAD system for the automatic detection of clustered microcalcifications in digitized mammogram films. *IEEE Trans. Medical Imaging*, 19(2):115–126, February 2000.

Computer Science Department, Babeş Bolyai University, Cluj Napoca, Romania

E-mail address: fratean.stefana@gmail.com

E-mail address: lauras@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

PERCEPTUAL EVALUATION OF RANDOM NUMBER SEQUENCES USING FILESEER+

KINGA MARTON, DAN PATRASCU, AND ALIN SUCIU

Abstract. The quality assessment of random number sequences based on visual perceptions is meant to complement the more widely used approach of statistical testing that has become a standard method of randomness evaluation. Instead of carrying out a well defined procedure by means of computer instructions which build-up the statistical randomness batteries, in visual assessment we rely on the perceptual system to extract statistical properties from the visual representation of the tested sequence and derive conclusions regarding the quality of randomness accordingly. FileSeer+ is a follow-up work to the research results previously presented in [1] and proposes several visual tests together with an efficient testing environment for random number sequences based on perceptual assessment. Empirical results show that visual evaluation can highly improve the process of randomness assessment by facilitating the understanding and interpretation of randomness through the properties of random sequences that can be visually captured.

1. Introduction

The quality assessment of random number sequences based on visual perceptions is meant to complement the more widely used approach of statistical testing that has become a standard method of randomness evaluation. Through visual assessment the tester can gain insight into the fascinating domain of randomness and randomness testing by understanding what randomness is and how it looks like by appealing to the perceptual instead of the cognitive system.

Received by the editors: January 21, 2015.

2010 *Mathematics Subject Classification.* 94A17, 68U99, 68U10.

1998 *CR Categories and Descriptors.* G.3 [**Probability and statistics**]: Random number generation – ; H.1.2 [**Information Systems**]: User/Machine Systems – *Human information processing*; H.4.2 [**Information Systems**]: Information systems applications – *Decision support*.

Key words and phrases. randomness assessment, visual perception, statistical properties, testing environment.

The human visual system is highly trained to extract features of the surrounding world and summarize these with statistical descriptors. This is due to the fact that our perceptual system is constantly concerned with selecting and organizing the sensory information with the intention of interpreting the outside world. This organizing tendency is so powerful that sometimes it slightly distorts the sensory information, hence sometimes we seem to perceive forms and shapes even when the received stimuli provide a very incomplete evidence of these. At the same time, our visual perceptions are also guided by our expectations of what we will see.

Depending on the nature of the entropy source, the generators can be classified in three categories, namely true random number generators (TRNGs), unpredictable random number generators (URNGs) and pseudo-random number generators (PRNGs). TRNGs extract randomness from a natural physical phenomenon, such as radioactive decay, thermal noise or quantum mechanics, and the properties of independence and unpredictability of the generated values are guaranteed by physical laws. URNGs are based on the unpredictability inherent to human computer interaction and on the undeterminism introduced by the complexity of the underlying phenomenon. PRNGs produce random looking sequences by expanding an initial value, called seed, by means of a deterministic recursive formula.

When applying statistical tests for assessing the randomness quality of a sequence, a finite set of statistical properties is evaluated based on values expected from a perfectly random stream. Instead of carrying out a well defined procedure by means of computer instructions which build-up the statistical randomness batteries, in visual assessment we rely on the perceptual system to extract statistical properties from the visual representation of the sequence and derive conclusion regarding the randomness quality of the sequence accordingly.

Furthermore, statistical tests may capture the lack of randomness but are limited in determining the cause or suggesting ways of readjusting the generators. Perceptual tests are more powerful in this direction, and hence are very useful especially in assessing physical devices (TRNGs) which may show certain disturbances but can be redressed if the cause is known.

This paper is a follow-up to the research results previously presented in [1], where we have highlighted some of the advantages and limitations of visual randomness evaluation and provided several relevant empirical results. The new contributions of the present work include the extended set of visual tests, the possibility to select and focus on a specific area of interest within the visual representation, the ability to generate random walks according to several rules in one and two dimensions. Furthermore, the testing process and its performance is highly improved through the integration of concurrent

working threads, allowing the generation and visualization of several representations concurrently. Moreover, the designed testing environment allows for easy and practical organization of the tested sequences, the associated visual representations, and all the generated results. Hence previous tests results can be saved, reloaded and extended through further analysis.

The paper is structured in 5 sections. Section 2 briefly presents the main difficulties in the process of randomness assessment, some scientific evidence which prove that visual perception can effectively contribute to abstract data analysis, and the main limitation in visual assessment, namely that visual randomness cannot guarantee real randomness. Section 3 points out the main contributions of the paper, presenting the included tests and the proposed testing environment, followed by experimental results in section 4, demonstrating the effectiveness of our approach. Section 5 presents final conclusions and further work.

2. Randomness assessment and visual perception

2.1. **Difficulties in randomness assessment.** Randomness assessment is a rather complex and resource expensive process characterized by peculiar properties, a few of which are briefly described in the following.

There is no finite amount of testing that can guarantee perfect randomness, but applying several relevant tests could increase the confidence in accepting the sequence as being random or rejecting it due to evidence of nonrandomness.

Selecting a relevant set of tests is difficult, because accurate results are expected from a relatively reduced number of tests and within a relatively short period of time. Hence a fair balance is needed between thoroughness and performance, though, no set of tests can be considered complete.

The most widely used randomness tests are statistical tests, which can be grouped together forming batteries of tests, such as the NIST statistical test suite [3], TestU01 [5] designed by LEcuyer and Simard, the Diehard test suite [9] developed by Marsaglia, and John Walkers ENT [7]. Statistical tests are usually based on hypotheses testing, summarizing their results in so called P-values, probability values between 0 and 1. Results may include a multitude of P-values, and are relatively hard to interpret by users unfamiliar with concepts of mathematical statistics.

A very important aspect which adds to the complexity of testing is the size of the tested sequence, considering that even perfectly random sequences contain subsequences that look deterministic and hence may fail some of the tests, yet the assessment process has to avoid reaching to an incorrect conclusion and consider the whole sequence as being nonrandom. At the same

time nonrandom sequences may contain subsequences which look random and consequently pass the majority of tests, yet the sequence as a whole may show repeating patterns, correlations or other traces of lack of randomness. Therefore the sequence to be tested has to be long enough to allow the evaluation process to arrive to the correct conclusion.

2.2. **Data analysis using human visual perceptions.** The theory and research in the domain of human perception and sensation has known very important advancements in the last few decades and continues to receive valuable contributions with rich experimental support. In our research we use several results from this field which prove that statistical information about a set of similar objects could be perceived by the perceptual system as accurately as the characteristics of a single object.

In particular, we rely on the work of Ariely [6] who studied the way a set of similar objects are perceived, and showed that the human visual system creates an internal representation of the overall statistical properties of the set, such as mean and distribution. Furthermore he experimentally showed (focusing on mean discrimination and member identification), that the visual system decides for which part of the set to retain the individual representation and for which to extract only the global statistical properties.

Chong and Treisman in [8, 4] are concerned with the way statistical properties are encoded and represented, and empirically measure and prove that our perceptual system is indeed highly capable of accurately judging the mean of items, with focus on the size and orientation property.

In [2] Robitaille and Harris provide evidence that the process of estimating the statistical properties by the visual system can highly benefit from enlarging the set of objects.

In the following we present how these results are adapted and used in our goal of assessing the quality of random number sequences.

In visual randomness testing the sequence of random numbers is graphically represented as one image or several images. Therefore the elementary object forming the random sequence in visual representation is a pixel, and the human perceptual system is required to extract statistical properties of the set of pixels which form the image, or a selected part of the image.

For a high quality random number sequence we expect to perceive uniform distribution and patternlessness in the image representation whether black and white, grayscale or color. The perceptual system identifies areas where the sequence seems to fulfill these requirements and creates an internal representation of the overall statistical properties of the area. Similarly, our visual system discriminates areas where there are visual traces of lack of randomness, such as certain (possibly repeating) patterns, bias towards a certain

value, or certain types of correlations. Furthermore, the human visual system is capable of capturing several statistical properties of the visual representation almost simultaneously.

The process of visual analysis can highly benefit from enlarging the visual representations by testing larger sequences of random numbers and by providing several different representations of the same sequence.

2.3. **Visual randomness.** The perceptual evaluation of random number sequences using the human visual system, in a way similar with every other statistical randomness tests, does not provide a method for proving randomness. Instead it takes advantage of our perceptual system to quickly spot tracks of predictability or non-randomness in the sequence and facilitates the understanding of randomness and lack of randomness.

Nevertheless, by representing a sequence of numbers graphically, the human visual system is only capable of determining the degree to which the representation satisfies visual randomness but is unable to tell the difference between real randomness and visual paternlessness.

In this context, visual analysis is not to be used exclusively, but rather as an efficient component of a larger randomness testing system which also includes powerful statistical test suites and other approaches to randomness evaluation.

3. The inspection tool: FileSeer+

FileSeer+ is a visual evaluation environment and is the result of a follow up work to the inspection tool named FileSeer, presented in [1]. The most important features are listed bellow with emphasis on the original contributions of the present evaluation environment.

Both tools provide the possibility to represent the tested sequence in three different image forms as: black and white (1 bit/pixel), grayscale (1 byte/ pixel) or color images (3 or 4 bytes/pixel) using BMP files. But while the original tool had a predefined dimension of the image in length and width, FileSeer+ allows the tester to adjust these parameters. Furthermore, FileSeer+ allows the visualization of several representations of the same sequence concurrently. These new features may significantly aid the visual system in spotting certain regularities and patterns and may facilitate the understanding of the underlying problem in the random number generator that produced the tested sequence.

The image representations of the sequence can be zoomed in for a more detailed visual inspection and FileSeer+ allows the selection of an interest area for further analysis.

Another completely new feature of FileSeer+ is the integration of several image filters which can be applied on the visual representation of the random sequences in order to emphasize certain deviations from randomness, especially bias and repeating patterns. The integrated filters include edge detection, mean and median filtering, blurring, sharpening and smoothing algorithms.

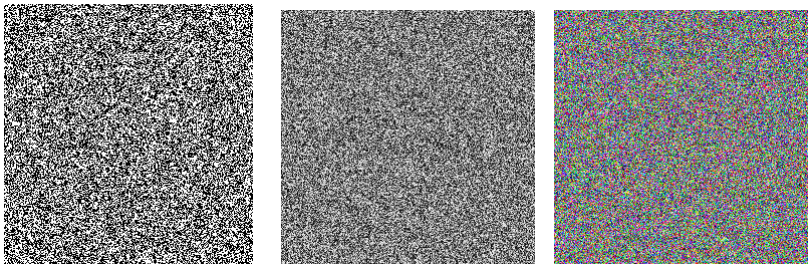
Another powerful component of FileSeer+ is the generation of random walks in one and two dimensions, in the latter case considering eight or four neighbors respectively. When generating random walks we expect a random sequence to show a tendency to fill the available space without showing repetitive patterns or rapidly leaving the area. Random walks can be drawn in black and white, grayscale or color and may be generated step-by-step or only the final result. Another important feature is that the visual area follows the current position in the random walk and it expends as the walk covers a larger space.

As we have already mentioned before, the human visual system is only capable of determining the degree to which the representation satisfies visual randomness but is unable to tell the difference between real randomness and visual paternlessness. Hence, in order to help the human analyzer in reaching to a correct decision on the quality of the tested sequences, certain statistical properties are visually represented, such as balance, entropy and histogram of values. Entropy is the numerical measure of uncertainty, usually expressed in bits per symbol (byte), hence the higher the entropy, the more difficult it is to predict the sequence. The balance expresses the extent to which 0 and 1 bits appear with the same probability and the histogram shows the number of occurrences for every one, two, four and eight bit values from the input file.

FileSeer+ is a visual testing environment providing a well structured working space where randomness testing projects can be created. Projects can be opened, modified, closed or removed and in each project files with random number sequences can be added or removed. For each file the tester may choose to create several representations, apply image filters, visualize statistical properties, generate random walks and save the desired results in the structure of the project for further use, significantly improving the testing process.

4. Experimental results

The visual randomness evaluation environment was tested using a large number (more than one thousand) of input files containing data of various level of randomness quality, which was also measured using well known statistical batteries for randomness assessment such as the NIST STS [3] and TestU01 [5].



(A) (B) (C)

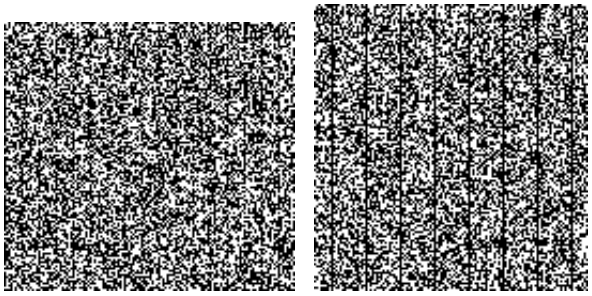
FIGURE 1. Image representation of a high quality random number sequence: (a) black and white image, (b) grayscale image, (c) color image

As we have already mentioned before, in the process of visual analysis we expect to perceive uniform distribution and patternlessness in the image representation whether black and white, grayscale or color - of a high quality random number sequence. Figure 1 shows the image representations for a high quality random number sequence generated by a true random number generator (TRNG).

The importance of allowing the tester to specify the image dimensions in length or width, is a simple but powerful feature because in case of input sequences which show a regular bias, certain dimensions of the image representation can ease the spotting of repeating patterns and nonuniform distribution more than others. Figure 2 depicts the image representations of such a sequence that has a severe bias of every 16th bit towards 0 values. The image on the right expresses more pronouncedly the existence of this bias that forms repetitive patterns.

While analyzing the image it is very useful to get a closer look to a selected area by zooming in. FileSeer+ provides five zooming levels that can be adjusted between one and five times the original size. Figure 3 depicts the color image representation of a larger sequence from the same input file as in the previous experiment, together with a three times zoom level of a selected area in which the colored strikes due to the above mentioned bias are more clearly visible.

Correlations between the subsequences within the input file and repetitions which can be due to exceeding a pseudorandom generators period or a reinitialization with the same seed may form visible traces, as shown in Figure 4, where the same input file is represented in both black and white and grayscale image respectively. There are several repeating patterns which can be easily



(A) (B)

FIGURE 2. Two black and white image representations for the same sequence with different image dimensions

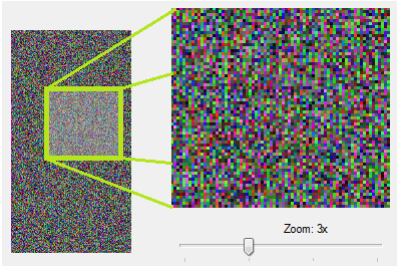
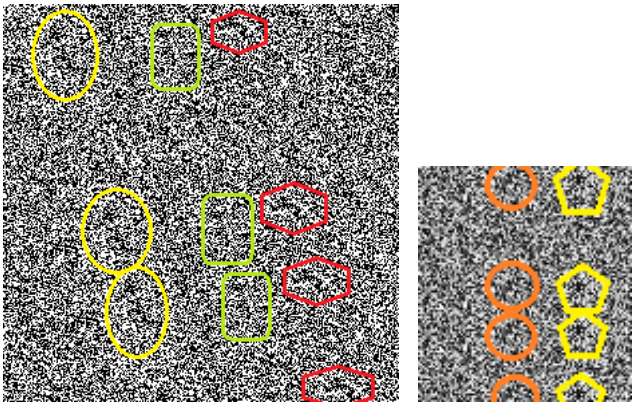


FIGURE 3. Color image representation of a biased sequence and a 3x zoom of a selected area

spotted and some of the most visible patterns are circled with different shapes and colors here to facilitate their tracking.

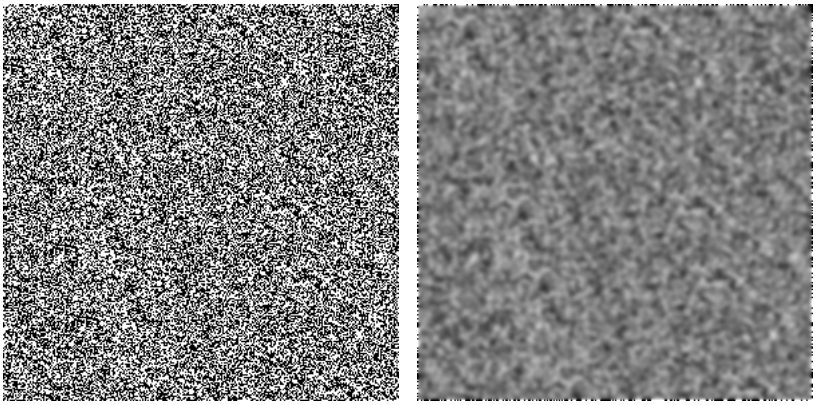
In order to aid the tracking of patterns and nonuniform distribution, FileSeer+ provides a set of image filters (several smoothing and sharpening filters and edge detectors) which can be applied on the visual representation of the sequence in any order and possibly multiple times, allowing also to undo any filtering step if the tester considers its effect unhelpful. Figure 5 exemplifies the usefulness of applying image filters on images when the existing patterns are not easily visible to the human eye. The original black and white image representing the tested sequence is provided on the left side and on the right we can see the effect of applying smoothing filters.

Another way to visualize the tested sequence is by generating random walks. FileSeer+ provides the possibility of creating one and two dimensional walks, and for the latter the user may select between considering four or eight



(A) (B)

FIGURE 4. Black and white and grayscale image representation of an input file with correlations and repetitions. Some repeating patterns are circled



(A) (B)

FIGURE 5. Smoothing filters for highlighting patterns: (a)original black and white image, (b) filtered image

neighbors in determining the direction of the next step in the walk. Figure 6 shows the encodings used for the two dimensional walks.

For a high quality random number sequence we expect the random walk to fill the available space without showing repetitive patterns or a tendency to rapidly leave the area. Figure 7 presents the random walk generated using four

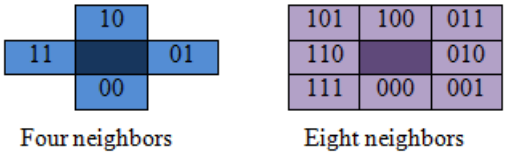


FIGURE 6. Encodings used for the two dimensional random walks

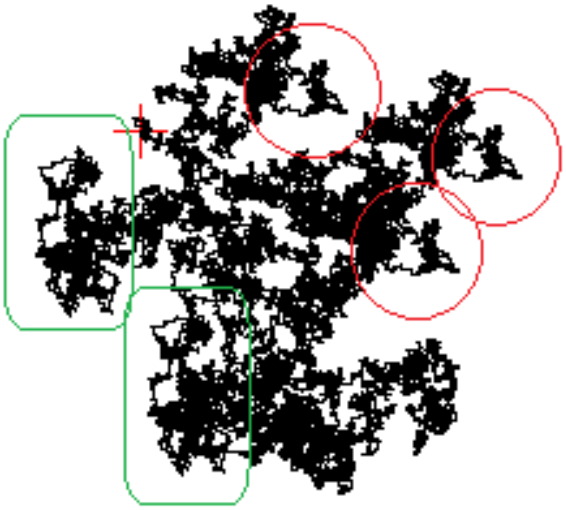
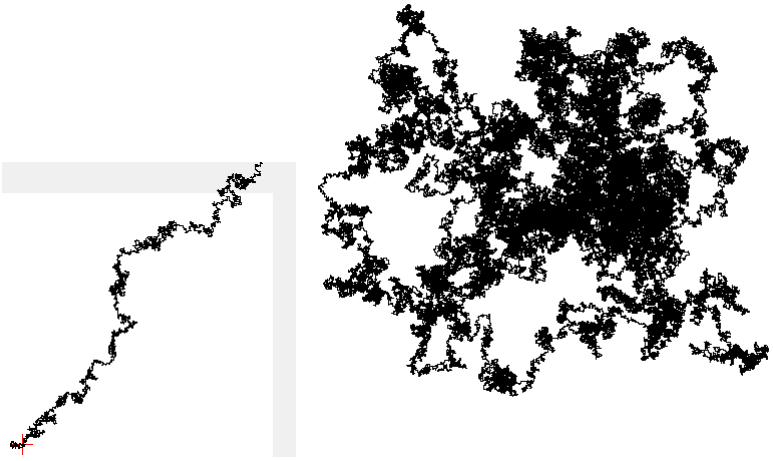


FIGURE 7. Random walk with 4 neighbors for a sequence containing repeating subsequences. The most visible repeating patterns are circled

neighbors and the most visible repetitive patterns are circled. These pattern show that the tested sequence contains recurrent subsequences or correlations. The red cross indicates the starting point of the random walk.

Another evidence of the lack of randomness is shown in Figure 8a, where the sequence is significantly biased towards certain values and hence there is a slight trace while leaving behind the available space, even if the considered sequence is very large. For comparison, Figure 8b depicts the random walk created by a high quality random number sequence.

The randomness evaluation based solely on the analysis of image representations has the serious limitation of not being able to discern between real randomness and visual paternlessness. Hence, before drawing a conclusion it is very helpful to visualize some very basic statistical properties, such as the



(A) (B)

FIGURE 8. Figure 8. Random walks: (a) biased sequence, (b) high quality random number sequence

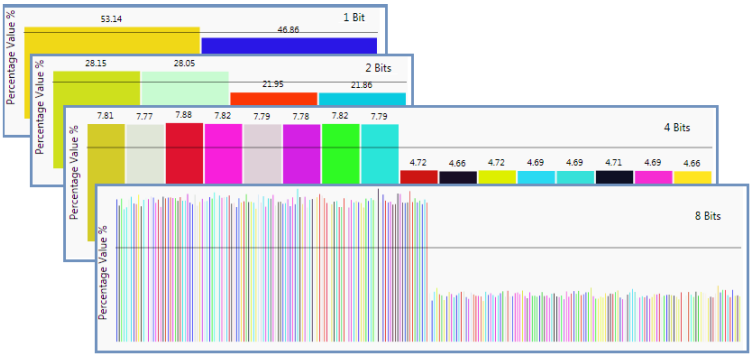


FIGURE 9. Histograms considering 1, 2, 4 and 8 bit values

frequency, the entropy or the balance of values, which can be rapidly computed and graphically represented for the whole sequence or for selected interest areas. Figure 9 presents the one, two, four and eight bits histograms for visually analyzing the frequency of values in the sequence represented as black and white image in Figure 2. The bias towards lower values is clearly visible.

The testing environment provides a highly organized work space where testing projects can be created grouping together test files and folders and

where all visual representations can be saved for further use and analysis. Furthermore, each visualization process is handled separately by a different thread so that several visual representations can be generated for the tested sequences helping the tester to compare the results and choose to record the most representative and relevant results.

5. Conclusions

In this work we have presented a testing environment for random number sequences based on visual perceptions. The domain of visual perceptions provides us with a large amount of scientific evidence which show that the human visual system is highly adapted to analyze large sets of similar objects and to extract individual or global statistical properties of the set. This ability is employed by our visual inspection environment in analyzing visual representations of the input sequences and based on the received visual perceptions help the tester to quickly decide whether to reject the sequence for evident traces of lack of randomness, or to further analyze the sequence with other assessment methods, such as statistical testing batteries.

The testing environment, named FileSeer+, provides several methods for visualizing the tested sequence such as black and white, grayscale and color image representations, (with the important feature of adjustable dimensions), the selection of an interest area, several levels of zooming, image filtering and random walks. These are complemented by the visualization of certain statistical properties of the sequence such as the one, two, four and eight bit histograms, the entropy and balance of values.

We have empirically shown how the human tester can use these methods to search for traces of nonrandomness in the visual representation of the tested sequence. Non-uniform distribution of values, the presence of repeating patterns, bias towards certain values and correlation within the sequence can be visually captured and the results demonstrate that the process of visual assessment is very efficient in complementing other more standard randomness evaluation approaches, such as statistical testing. Furthermore perceptual analysis has the major advantage of facilitating the tester to gain insight into the fascinating domain of randomness and randomness testing by understanding what randomness is and how it looks like by applying to the perceptual instead of the cognitive system.

FileSeer+ is not just a tool, but a complex testing environment which allows the tester to create testing projects and efficiently organize the working space. The tester can generate several different visual representations for the tested sequences, keep these representation in the working space for comparative analysis and may choose to save some or all the representations

and associate them with the sequence within the created testing project as evidences or for further analysis.

We aim to further improve the testing environment by extending the provided visual representation methods, design an automated method for recognizing problematic areas, and integrate parallel processing for a more efficient assessment of input sequences.

Acknowledgments

This paper was supported by the Post-Doctoral Programme POSDRU /159/1.5/S/137516, project co-funded from European Social Fund through the Human Resources Sectorial Operational Program 2007-2013.

References

[1] D. Ariely, *Seeing sets: representation by statistical properties*, Psychological Science, 12 (2001), pp. 157-162.

[2] S. C. Chong, A. Treisman, *Representation of statistical properties*, Vision Research, 43 (2003), pp. 393–404.

[3] S. C. Chong, A. Treisman, *Statistical processing: Computing the average size in perceptual groups*, Vision Research, 45 (2005), pp. 891-900.

[4] P. L'ecuyer, R. SIMARD. *TestU01: A C Library for Empirical Testing of Random Number Generators*, ACM Trans. Math. Software, Article 22.

[5] G. Marsaglia. *The Diehard test suite*, 2003, Available online: `http://www.csis.hku.hk/diehard/`

[6] K. Marton, I. Nagy, A. Suciu, *Visual Inspection of Random Number Sequences with FileSeer*, Automat. Comput. Appl. Math., 19/1 (2010), ppl. 3–10.

[7] N. Robitaille, I. Harris, *When more is less: extraction of summary statistics benefits from larger sets*, Journal of Vision, 11/12 (2011), pp. 1–8.

[8] A. Rukhin et al. *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, NIST Special Publication 800-22 (with revisions dated April, 2010).

[9] J. Walker. *ENT - A Pseudorandom Number Sequence Test Program* . Fourmilab, 2008, Available online: `http://www.fourmilab.ch/random/`

Computer Science Department, Technical University of Cluj-Napoca
E-mail address: {`Kinga.Marton, Dan.Patrascu, Alin.Suciu`}`@cs.utcluj.ro`

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LX**, Number 1, 2015

A DECISION SUPPORT SYSTEM FOR COLOR MATCHING IN DENTISTRY

VASILE PREJMEREAN, VASILE CIOBAN, ALEX GHIRAN, DIANA DUDEA, AND BOGDAN CULIC

Abstract. Color is very important in oral medicine. This article presents a few solutions to problems that arise in dental image color analysis. Images taken by a digital camera are calibrated (using *VitaPan 3D Master* and *18% Grey Card*) and, afterwards, the colors that appear in various areas of interest are analysed. The user interface uses the CIE $L^*a^*b^*$ color space and for color comparison the ΔE^*_{ab} distance metric is used. The aim of this article is the development of software capable of color analysis performed on a digital image of a dental tooth structure or restorative surface. The analysis of digitally recorded images, performed by computers, may provide important information that can help dentists in achieving superior results.

1. Introduction

The instrumental methods for dental color analysis have been introduced in practice in order to transform a subjective analysis into an objective method, which allows the numerical expression, through different systems, of dental color parameters. As a result, variations induced by particularities of individual perception are avoided, as well as the errors generated by the phenomenon of metamerism.

In the current climate of dentistry the most used tooth shade selection methods are subjective. The methods depend on the observer, who has to compare tooth color with selected shade tabs from different shade guides.

Received by the editors: December 10, 2015.

2010 *Mathematics Subject Classification.* 68U35, 65D18, 68U10.

1998 *CR Categories and Descriptors.* H.4.2 [**Information Systems Applications**]: Types of Systems — *Decision support*; I.4.6 [**Image Processing and Computer Vision**]: Segmentation — *Region growing, partitioning*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis — *Color*; J.3 [**Computer Applications**]: Life and Medical Science — *Health.*

Key words and phrases. Tooth color, image processing, visualization, medical imaging, computer technology, dental informatics, digital x-ray and imaging devices, computer based oral health record, decision support systems.

The recent technological improvements in the area of computers, communication networks and the Internet have greatly affected contemporary society [10], [11]. These improvements have translated in an ongoing improvement of dental medicine. A new generation of technologies focused on the analysis, communication and color checking were developed in recent years. Spectrophotometry, Colorimetry and Digital Image Computer Analysis are the instrumental methods used in dental practice and research ([2], [3]). The instrumental dental color selection methods require the purchase of expensive equipment, which may require specialized training, and which is generally outside the available means of dental laboratories and surgeries, both in Romania as well as abroad. As result, these techniques are rarely used in current dental practice largely relegated to research laboratories and academia.

Digital analysis of dental images must be taken into account, not only when information related to dental shade needs to be transferred from the dentist to the dental technician, in order to reproduce the optical properties of dental structures using esthetic dental materials, but also when color parameters need to be recorded in order to monitor the changes in dental shade generated by some extrinsic factors (for instance, tooth staining or dental bleaching). The use of commercial digital cameras for accurate color capture is advantageous, but in order to be relevant to clinical research color difference parameters must be defined [12]. Digital methods of color analysis have become more widespread lately, their usage being extended into dental research, mainly in order to follow the results of a treatment which involves the change of dental shade [8].

The development of easy to use open source software for dental color matching, aimed to generate predictable results, is likely to improve the performances in color selection for both clinicians and dental technicians.

2. Program functions

In this section the main functionalities of the application are described.

The first category of functionalities that must be performed by the application is related to the use of images. The functionalities are the loading, saving, clearing of images and color key generation corresponding to the 26 shade tabs used.

The second functionalities category is related to calibration (see Figure 1. level 1 – Calibration). This functionalities category is important due to the fact that cameras when taking pictures alter the real colors. The calibration functionalities category consists of two different calibration types: calibration using shade tabs and calibration using 18% Grey Card.

The third functionalities category refers to the way in which the application's analyzed area is specified (see Figure 1, level 3 – Study area). The area may be specified either automatically or manually by the user. From the analyzed area the application must be able to exclude color anomalies, such as those created by flash light reflections.

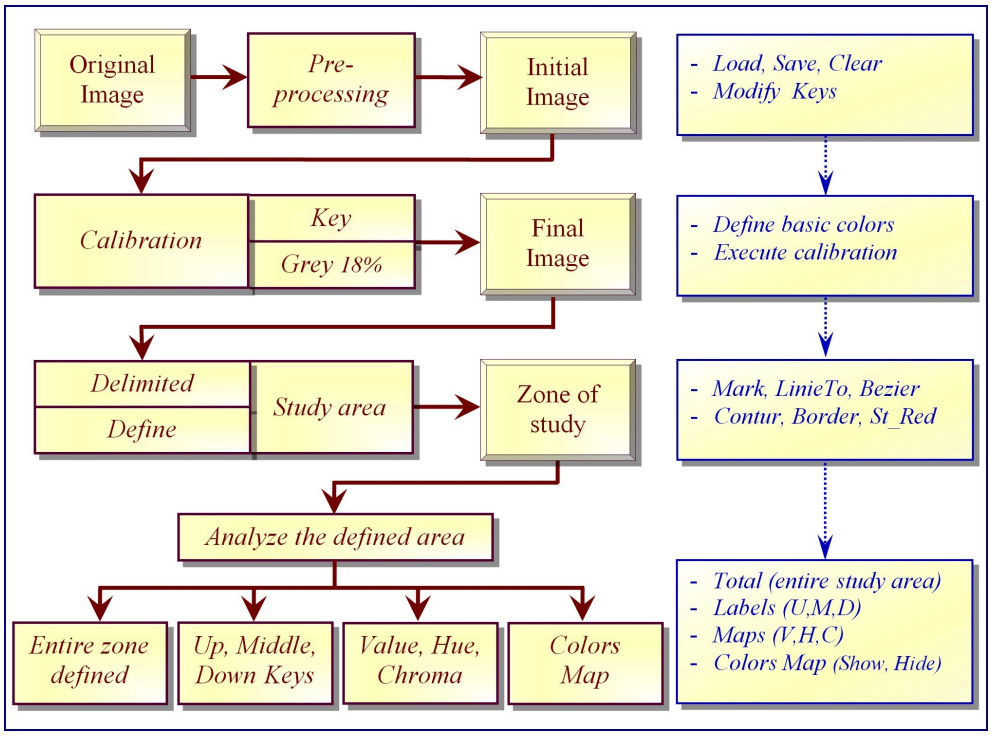


Figure 1. Color analysis features and steps

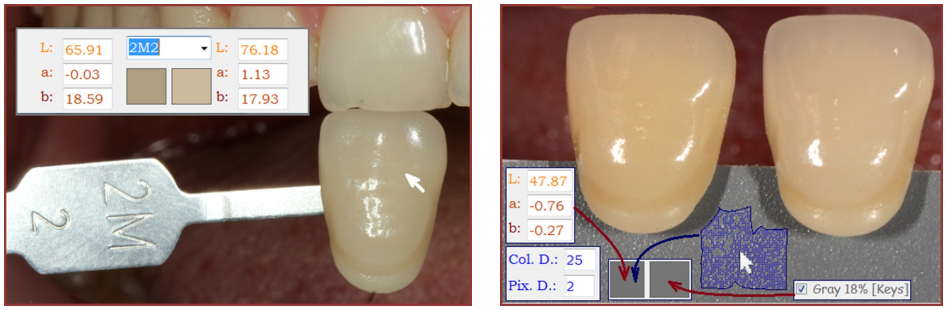
The final functionalities category is related to the application's results. The application needs to have two color analysis options: a global color analysis for the entire tooth area and a tooth zone analysis, corresponding to the cervical, middle and incisal areas in which the tooth contour was divided.

3. Implementation

The digital camera alters the colors which can lead to a wrong choice of color keys. For this reason, image calibration (the second functionality category described in Section 2 "Program functions") must be performed before proceeding to the proper analysis of the tooth's color (see Figure 1). Before color study can be performed, a necessary step is the determination of the

image area corresponding to a tooth. Color analysis ignores undesirable areas (reflection or transparency areas) and computes the color search key which will be used from then on.

The first processing step performed on the dental image is color calibration (see Figure 1), performed in order to improve the realism of the image (see [4]). A first possible method is based on knowledge of a color key contained by the image. In practice, two colors are defined: the initial color of the base area of the image, denoted by C_{Init} in equation (1), computed as an average of the area's colors, and the final color associated with the color search key used, denoted by C_{Final} in equation (1) (see Figure 2a). The application user specifies the center of the initial area and is able to modify the implicit area parameters (the pixel network density as well as the color threshold used in computing the borders of the area that will be used to compute the color average of the area). The color search key used for calibration will be specified by selecting it from a list.



a) Using 2M2 Key b) Using 18% Grey paper

Figure 2. *Initial* and *final* calibration colors definition

Modifying image colors will only be performed after a preprocessing stage that makes sure that the color template is in correspondence to the real color. The transformation, performed on each color component (red, green and blue) C, is given by the following formulae:

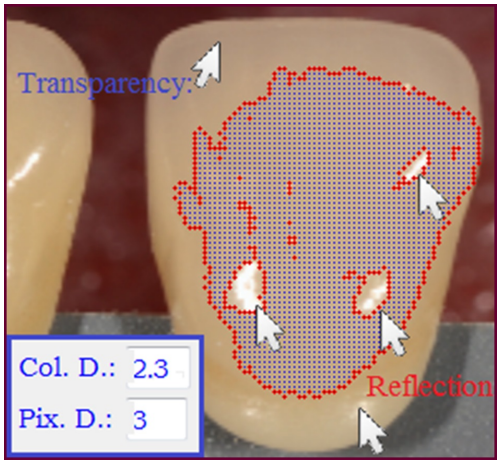
$$C' = \begin{cases} C * C_{Final}/C_{Init} & \text{if} \quad C * C_{Final}/C_{Init} \leq 255 \\ 255 & \text{if} \quad C * C_{Final}/C_{Init} > 255, \end{cases} \tag{1}$$

where C_{Init} and C_{Final} denote the initial, respectively the final calibration colors.

The second implemented method makes use of an object in the image whose color is known and fixed. An 18% gray card made of paper for which the real values $(118, 118, 118)$, in the red green blue (RGB) color space, are

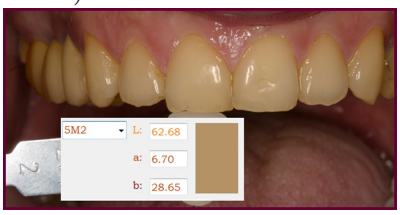
known was used. The formulae used in the transformations performed by this method are the same ones used by the first described method (see Equation 1). The definition area of the initial color portrayed as blue in Figure 2b is specified by the application's user using the mouse cursor directly on the image in order to select the starting pixel coordinates. The user is also able to modify the implicit values for the accepted color distance and for the distance between pixels. The final color obtained after calibration is Grey 18%. Testing has determined that the results obtained by this calibration method are much poorer than those obtained with the first calibration method described. This may be due to the fact that the 18% grey color is not present in the dental color subspace, whereas the final color obtained using the first calibration method is much closer to real teeth colors. Nevertheless, this calibration method is useful due to its significant benefits in automatic calibration. Regardless of the calibration method used, the colors obtained after calibration are much closer to the real colors than the uncalibrated image colors were.

The next processing step consists in the determination of the studied area (the third functionality category described in section 2 "Program functions") where color analysis will be performed. The distance metric ΔE^*_{ab} between two colors is compared to the color acceptability threshold (which usually has a value around 2.5) and if the distance is greater than the threshold the colors are considered distinct, otherwise they are considered to be similar (see Figures 3a and 3b, where the threshold is denoted by "Col. D"). This threshold has an implicitly defined value in the application but it can also be modified by the user.



a) Marking studied area

b) Before modification

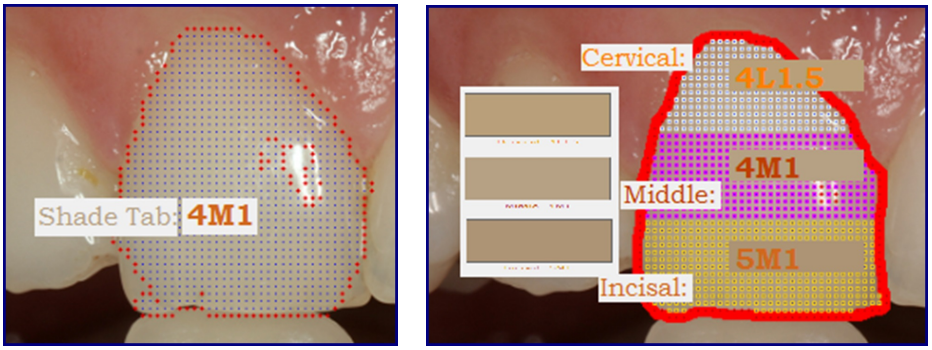


c) After modification

Figure 3. Contur determination

If a smaller area is desired the user has at his disposal several options for defining the tooth's border directly on the image using pixels, lines or curves (*Mark*, *LineTo* or *Bezier*). The algorithms used for marking the studied area (*Contour*, *Border*, *St_Red*) are filling algorithms (*FloodFill*) they are non-recursive due to the fact that recursive variants were discovered to give much poorer results on large areas (such as those in Figure 3b and 3c). The algorithms are adapted to the application's functionalities and after the determination of the studied area, the color spots, present due to reflection or transparency, are excluded (see [5], [6]).

The result can be given by the average taking into account the entire marked area or only the three interest areas (see Figure 4a). In order to obtain the shade tab for the entire area defined, the average color will be compared with the previously determined color key pattern (standard palette – *banana* in the color space that represents tooth color subspace), and the result will be given as the closest color code as is used in dentistry (see [9]). Tooth color is given by light reflection that passes through the enamel and is reflected from the adjacent dentin layer. It presents an uneven color distribution on the tooth surface: the cervical area (the area next to the gingiva) is usually more saturated and opaque; the middle area presents a more even color distribution but colored texture can also be present at this level; the incisal area contains more thin enamel and no dentin layer, with a high degree of translucency. In the images this areas will appear black, because of the translucency. For the three interest areas (the *cervical*, *middle* and *incisal* areas) the average colors are computed, then for each the corresponding shade tab is computed and for each of them the code and hue is displayed (as it can be seen in Figure 4b).



a) Total – for entire zone b) Labels – cervical/middle/incisal

Figure 4. Shade tab

The obtained results are expressed in Vita Pan 3D Master shade guide. It is the most widely used shade guideline, containing 26 shade tabs, evenly

distributed in dental color space [7]. Shade tab code (e.g. 2M1) are displayed by the program on a label placed on the tooth surface in the middle area, for the global analysis mode, or by three labels, each in its corresponding area, for tooth zone analysis mode. When a color of the dental surface is selected, an exact match of the shade tab is never possible. It will always be a color difference between the selected dental color and standard shade tabs. The software calculates ΔE^*_{ab} values between the L^*, a^*, b^* values measured on the digital image and shade tabs corresponding parameters (already implemented in the program). The final result is the shade tab that generates the lowest ΔE^*_{ab}.

When studying the resulting colors (hues) the user has the possibility to study the hue difference on the surface of the tooth on each of the coordinate of the HSB color space (*Hue*, *Saturation*, *Brightness* alternatively called HSV: *Hue*, *Saturation*, *Value* [1]) as it can be seen in Figure 5.

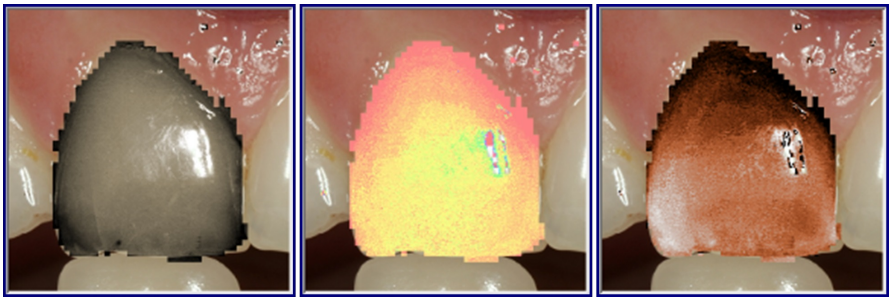


Figure 5. HSB coordinate color view

A tooth color map is also displayed. In order to notice the hue differences the colors will be codified (falsely colored) and the most frequent keys (i.e. the first 11 keys) will be displayed in the legend (see Figure 6). The other colors, that are those with low frequencies, are represented using the white color (*Other*).

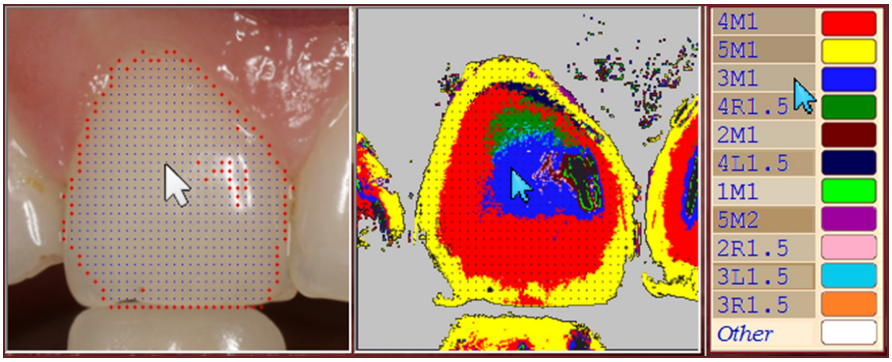


Figure 6. Tooth Color Map

4. Conclusions and further work

Color matching in dentistry is a complex process, which can generate errors, regardless of the method involved in shade selection (either visual or instrumental).

An experimental software for color selection was developed in the presented research. The program was able to match the shade tab colors, from calibrated digital images.

An experimental study was made; testing the accuracy of the program in measuring shade tabs color. Digital images of shade tabs were taken, and colors were match using the software.

The program was able to match the shade tab colors; the statistics indicates good accuracy when the results were presented in Vita 3D Master codification. A very strong correlations of the results were obtained Spearman's rank correlation = 0.914 ($p < 0.001$).

However, 19.23% of ΔE^*_{ab} values obtained were above the reference value of 3.2.

Further in vitro and in vivo evaluation of the program needs to be done in order to be used in routine clinical color selection. A tooth is usually polychromatic multiple colors can be found at the same time in different areas.

Further improvements of the program are indicated in order to be used for clinical color matching.

Acknowledgement. This study was supported under the frame of European Social Found, Human Resources Development Operational Program 2007-2013, project no. POSDRU/159/1.5/S/138776.

References

[1] D. Brigs, *The Dimensions of Colour*, http://www.huevalue chroma.com/index.php, Modified December 17, 2012.

[2] S.J. Chu, A. Devigus, A. Mieleszo, *Fundamentals of color: shade matching and communication in esthetic dentistry*, Tokyo, Quintessence Publ. Co, 2004.

[3] S.J. Chu, R.D. Trushkowsky, R.D. Paravina, *Dental color matching instruments and systems. Review of clinical and research aspects*, J. Dent., 38(2010), Suppl. 2., 2-16, Epub. Aug. 2010.

[4] T. Hayashi, *Medical Color Standard*, V. Tooth Crown, Tokyo, 1967, Japan, Color Research Institute.

[5] E. Hodneland, *Digital Images*, Cand. Scient. Thesis in Applied Mathematics, Image Processing Group (BBG), Department of Mathematics and Neuroinformatics and Image Analysis Group Department of Physiology, University of Bergen Segmentation of 22nd July 2003.

[6] O.J. Tobias, R. Seara, Member, IEEE, *Image Segmentation by Histogram Thresholding Using Fuzzy Sets*, IEEE Transactions on Image Processing, 11(2002), No. 12.

[7] R.D. Paravina, J.M. Power, *Esthetic color training in Dentistry*, Mosby, St. Louis, 2004.

[8] L. Schropp, *Shade matching assisted by digital photography and computer software*, J. Prosthodont. 18(3)(2009), 235-41, Epub. 2008 Dec. 30.
[9] R.C. Sproul, *Color matching in dentistry. Part I. The three dimensional nature of color. Practical applications of the organization of color*, Journ. Prosthet Dent, 5(2001), 453-457.
[10] H. Umar, *Capabilities of Computerized Clinical Decision Support Systems: The Implications for the Practicing Dental Professional*, The Journal of Contemporary Dental Practice, 3(2002), No. 1, 27-42.
[11] S. Wang, *A computer-aided analysis on dental prosthesis shade matching*, International Conference on Biomedical Engineering and Informatics (BMEI), 4th International Conference, 4(2011), 1950-1954.
[12] A.G. Weea, D.T. Lindseyb, S. Kuoa, W.M. Johnston, *Color accuracy of commercial digital cameras for use in dentistry*, Dent Mater, 22(6)(2006), 553559.

Babeş-Bolyai University
Faculty of Mathematics and Informatics, Department of Informatics
Cluj-Napoca, Romania
E-mail address: per@cs.ubbcluj.ro

Babeş-Bolyai University
Faculty of Mathematics and Informatics, Department of Informatics
Cluj-Napoca, Romania
E-mail address: vcioban@cs.ubbcluj.ro

2014 Graduated Technical University of Cluj-Napoca
Faculty of Automatic Control and Computer Science
Cluj-Napoca, Romania
E-mail address: ghiran_alex_cluj@yahoo.com

University of Medicine and Pharmacy "Iuliu Haţieganu" Cluj-Napoca
Faculty of Dental Medicine, Department of Prosthodontics
Cluj-Napoca, Romania
E-mail address: ddudea@umfcluj.ro

University of Medicine and Pharmacy "Iuliu Haţieganu" Cluj-Napoca
Faculty of Dental Medicine, Department of Prosthodontics
Cluj-Napoca, Romania
E-mail address: bogdanculic@yahoo.com