BIULETYN INSTYTUTU SYSTEMÓW INFORMATYCZNYCH

KOLEGIUM REDAKCYJNE
prof. dr hab. inż. Andrzej Ameljańczyk
dr hab. inż. Ryszard Antkiewicz
prof. dr hab. inż. Marian Chudy (redaktor naczelny)
dr Lucjan Kowalski (redaktor statystyczny)
dr hab. inż. Andrzej Najgebauer
dr hab. inż. Tadeusz Nowicki
dr hab. inż. Bolesław Szafranski
dr hab. inż. Zbigniew Tarapata (redaktor tematyczny)
prof. dr hab. inż. Andrzej Walczak (redaktor tematyczny)
dr hab. inż. Kazimierz Worwa

ADRES REDAKCJI
Redakcja Biuletynu Instytutu Systemów Informatycznych
Wydział Cybernetyki Wojskowej Akademii Technicznej
00-908 Warszawa, ul. S. Kaliskiego 2
e-mail: biuletynisi@wat.edu.pl
tel.: (22) 683-95-04, fax: (22) 683-78-58
http://biuletynisi.wcy.wat.edu.pl

Biuletyn Instytutu Systemów Informatycznych jest czasopismem indeksowanym w bazach:
http://baztech.icm.edu.pl/
http://indexcopernicus.com/

Redakcja techniczna i projekt graficzny okładki: Barbara Fedyna

Wydawca: Instytut Systemów Informatycznych Wydziału Cybernetyki WAT

ISSN 1508-4183

Biuletyn Instytutu Systemów Informatycznych jest czasopismem wydawanym dwa razy w roku.
Wersją pierwotną (referencyjną) czasopisma jest wydanie papierowe.

Druk: P.P.H. Remigraf Sp. z o.o., ul. Dźwigowa 61, 01-376 Warszawa
SPIS TREŚCI

1. M. Kniotek
   Early exit from a function as method of code optimization in the component environments ................................................................. 5

2. P. Michałowski
   Multi-criteria models of player’s preferences in investment process ................... 15

3. R. Pełka
   A multifaceted model for software reliability prediction during testing .............. 25

4. Ł. Ślezak, J. Butanowicz
   Security of Statistical Databases as an element of Enterprise Security Architecture ...... 37

5. K. Tomaszewski
   Clustering method based on the analysis of the access request stream in object-oriented databases ................................................................. 45

INFORMACJE DLA AUTORÓW ................................................................. 51
Early exit from a function as method of code optimization in the component environments

M. KNIOTEK
kniotek@agh.edu.pl

AGH University of Science and Technology,
Faculty of Geology, Geophysics and Environment Protection,
Department of Geoinformatics and Applied Computer Science,
30 Mickiewicza Av., 30-059 Kraków, Poland

This paper presents the use of code optimization method in the component environments (Java VM, MS .NET, Mono). In the proposed solution implemented for component environments there are occurrences when manual code optimization can accelerate run time of programs. In components environments, due to the JIT (Just In Time) optimization, the code cannot be fully optimized because of the short time available. JIT optimization takes place during execution of the currently used part of the code. That is why the time spent on searching the best optimization must be balanced between the user waiting time and the choice of optimal optimization. This article presents optimization method ending with conclusion on when, why and in which component environment is recommended to use a given method. The presented methods is called early exit from function (authorial method).

Keywords: code optimization, component environments.

1. Introduction

Component environments are widely used for coding multiple programs. One of their main advantages is code portability between different operating systems and computer architectures. The second advantage is the standardized notation of the code, which allows for using universal optimization methods [1].

This article was created in reference to [2], to show that the use of manual code optimization allows to achieve additional speed as opposed to using only optimizers incorporated in the component environments. Multiple programs may have long execution time. In this case, it is natural to search for the appropriate code optimizations that reduce the computation time. Authorial method presented in this article (early exit from a function) can accelerate run time of programs.

There are two cases where optimization should be applied: programs whose running time is long, such as used in the geophysics (inversion of plane-wave seismograms) [3], or in the multimedia, where results must be delivered quickly to maintain fluency [4]. Optimization methods are currently looked for in many areas of science, such as secrecy capacity of a network [5], mining signatures from event sequences in healthcare data [6], efficiency of the halo current diagnostic system foreseen in ITER (International Thermonuclear Experimental Reactor) [7], overall network throughput optimization [8, 9], code in the hat-fed reflector antenna [10], transmission of pre-encoded JPEG2000 video over time-varying channels [11], enhancement of the system performance of cellular network [12], processing of digital pictures [13], traffic distribution algorithm for network [14].

Optimization method presented in this article was tested. Method was performed on a different computer architecture and in different operating systems. In each case, program execution was timed, intermediate code examined, and results summarized in order to answer when, why and in which component environments it is recommended to use a given method. In most cases, after using presented optimization method the programs execution time decreased, but there are certain exceptions described in the results.

At the beginning of the article is presented compilation process with introduction to the optimization methods. After this is described authorial method of optimization called early exit from a function. In the next chapters are presented methodology, tests and conclusions about authorial method of optimization.
2. Compilation

Compilation in all the environments mentioned in this article is performed in the same way. In the beginning, the source code is compiled through compilers provided with the package for developers in the intermediate code. The obtained files are then compiled once again by a compiler in virtual machine at the program’s startup. As a result is obtained machine code which is executed by the processor. Compilation of the intermediate code to the machine code is performed by the JIT compiler, it takes place in stages and is subject to ad hoc optimizations at intervals set by the machine (thus the name: Just-in-Time).

JIT compilation occurs at the start of the application. It is performed at every launch of the program. The purpose of the aforementioned compilation is to translate the intermediate code into the machine code of the currently used platform. Each method is compiled only when there is need for it. Thus, the program can run without being fully compiled, because some methods may be unused. During the execution of the program, once the compiled parts are not lost and can be reused, they are loaded into the cache as ready to use the machine code [15, 16, 17].

JIT compilation is limited only by one factor: time. This is due to the fact of its execution during the launch of the application. Therefore, the analysis and code optimization cannot last as long as they could at the AOT compilation [18]. However, its advantages are code portability and application optimization suited to the currently used hardware and operating system. To compare, applications compiled using the JIT techniques run faster than scripts which are executed by interpreters [19].

3. Optimization methods

Programs can be implemented in many different ways and in all cases the result will be correct. However, certain approaches can be:

- easier (the solution of the problem itself thanks to which the implementation can be easier and more understandable)
- cleaner (they use less memory)
- easier to maintain (to adapt the code to frequent changes and improvements)
- faster (time to obtain the result is shorter) [20].

4. Early exit from a function

Early exit from a function is an method based on returning values from functions as fast as possible. Programmers should create as many as possible returning points in areas where we already know what will be the final return value of the executed function.

An example of a non-optimized code that stores resulting value in the variable result is shown in Table 1 (before optimization). After removing the variable result and replacing it with a few returning points it may be expected to use less memory and to accelerate the program (Table 1 – after optimization).

<table>
<thead>
<tr>
<th>Tab. 1. Early exit from a function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Before optimization</strong></td>
</tr>
<tr>
<td>int compare (int x, int y)</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>int result = 0;</td>
</tr>
<tr>
<td>if( x &lt; y ) {</td>
</tr>
<tr>
<td>result = -1;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>else if ( x &gt; y) {</td>
</tr>
<tr>
<td>result = 1;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>return result;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td><strong>After optimization</strong></td>
</tr>
<tr>
<td>int compare (int x, int y)</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>if( x &lt; y ) {</td>
</tr>
<tr>
<td>return -1;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>else if ( x &gt; y) {</td>
</tr>
<tr>
<td>return 1;</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>return 0;</td>
</tr>
</tbody>
</table>

5. Methodology and test performance characteristics

In order to distinguish test environments, later in the paper they are named by shortcuts E64 and E32. Numbers designate operating systems (32 and 64 bits) installed in the respective test environments.

The first test environment – E64:
Hardware specification:
- CPU: Intel® Core™ 2 E6400 @ 2.46 GHz
- RAM: 1.5 GB DDR2-667 (640 MHz).
Operating systems:
- Microsoft Windows 7 Professional SP1 (64 bit)
- Fedora 16 (64 bit).
The second test environment – E32:
Hardware specification:
- CPU: Intel® Celeron® M520 @ 1.60 GHz
• RAM: 512 MB DDR2-667 (532 MHz).

Operating systems:
• Microsoft Windows XP Professional SP2 (32 bit)
• Fedora 11 (32 bit).

Virtual machines installed in an E32 and E64 environments:
• Microsoft .NET 4.0 (Windows)
• Microsoft .NET 2.0 (Windows)
• Java Development Kit 7u4 (Windows and Linux)
• Mono 2.10.8 (Windows)
• Mono 2.4.3 (Fedora 11 – E32)
• Mono 2.10.5 (Fedora 16 – E64).

In order to reach objective test results, all testing was performed in the same way:

1. Tested instructions were carried out in a loop; the total time of their operation was measured. The loop had a predefined amount of iterations.

2. If the test required sample data, they were randomized for each iteration of the loop. Randomizing took place before the measurement of time and the values were stored in arrays. Data were randomized in order to keep the conditions close to real application run. Thanks to randomization, the data were both pessimistic and optimistic (those that can make to return result faster or slower). The randomizer was initiated using current time.

3. The measurements were performed for each test ten times. It is always the shortest time of all that was chosen. This approach is burdened with the smallest error, due to the applications that run in the background in a test environment [21].

4. To measure time with nanosecond accuracy, methods provided together with executing environments were used. In Mono and .NET, the Stopwatch class from System.Diagnostics package was used, while in Java, it was the System.nanoTime() method.

5. The use of the source code compilers to byte code:
   – .NET, compiler csc with /o flag (optimization launch)
   – Mono, compiler mcs with optimize+ flag (optimization launch)
   – Java, compiler javac (by default, optimization is turned on).

6. The use of JIT compilers:
   – .NET, lack of interference in the applied optimization
   – Mono, compiler mono with flag –O=all (turn on all possible optimization)
   – Java, compiler java with flag –XX:+AggressiveOpts (optimizations foreseen for the next release of JVM), also separately launched in mode –client and –server.

6. Performance of the early exit from a function

The test is executed to compare the time spans after applying optimization of the early exit from a function (4. Early exit from a function) and inline expansion. The test foresees cases when optimization is applied manually and the lack of their manual use. Both optimizations are presented in all possible variants of their use. In addition, the test is designed to draw attention to memory usage and intermediate code after disassembling.

A test version without the use of inline expansion was presented with the pseudo-code in table 2. The condition upon which the function printOnScreen(result) is executed is noteworthy here. In testing, this condition was never fulfilled, but using it ensured that the optimizer does not recognize the result variable as unused in the code. Optimizers, when encountering unused variables in the code, often use additional optimizations, which impacted the test. Typically, fragments of the code handling such variables are treated as redundant, while this fragment of the code is the critical part of the test. The lack of such condition made the measured times shorter. Two variants of the comparison function have been foreseen, namely before the use of optimization of the early exit from a function (labeled as A1 – Table 3) and after the optimization of the early exit from a function (labeled as A3 – Table 4). Variant A2 (Table 5) is an extension of the A1 variant (Table 3), namely there has been used the inline expansion. Variant A4 (Table 6) is a combination of variants A2 (Table 5) and A3 (Table 4), because both the optimization of the early exit from a function and inline expansion have been applied there.

Tab. 2. Pseudo-code for the early exit from a function test without the use of the inline expansion

```java
x[] = randomArray();
y[] = randomArray();
startTime();
for ( i = 0 ; i < 1 000 000; i++) {
    result += compare(x[i], y[i]);
}
stopTime();
if(result > 500 000)
    printOnScreen(result);
```
In Table 3, all the times measured during the test have been gathered. In addition, the acceleration after the use various optimizations (variants) has been calculated.

The measured times in the test environment E64

The fastest is the program executed in the .NET Framework and in the JVM, client version in Windows 7. It may be noted that the program in Fedora 16 is slower than the one executed in Windows 7. In Fedora 16 operating system, the slowest is the program executed in JVM, client version. While the fastest is the program executed in Mono, but the values are within the range of the slowest program execution in Windows 7. Time differences within each runtime environments and operating systems are more visible in the accelerations.

The measured times in the test environment E32

The fastest is the program executed in the .NET Framework and in the JVM, client version (both Windows XP and Fedora 11). The rest of the runtime environments are characterized by similar times of the program execution. In general, the programs executed in the test environment E32 have longer execution times compared to the test environment E64. This was due to the fact that the computer running the test environment E32 has weaker hardware parameters. For example, the program execution time in the runtime environment .NET Framework 2.0 is nearly twice longer. However, it is worth noting that in some cases this difference is smaller, for example the program in the server version of the JVM (within different versions of an operating system Fedora) executed in about 28% longer. It is interesting that the program in the JVM, client version (within different versions of the Fedora operating system) executed faster in the test environment E32. Overall, the program executed in the E32 environment in the JVM, server version is slower than running in the JVM, client version. Mono runtime has similar time to the JVM, server version. Time differences within each runtime environments and operating systems are more visible in the accelerations.
The acceleration in test environment E64

The acceleration in test environment E64, obtained after the use of three variants listed in the test (A2, A3, and A4) compared to the variant A1 which is not optimized. In some cases, the acceleration could not have been achieved. The program runs slower in the A2 variant in the runtime environment .NET Framework 4.0. The slowdown is also observed after the use of the A3 variant in .NET Framework 4.0 and the JVM client version in the Fedora 16 operating system. In all the other cases there were always accelerations. Acceleration happened even in cases with the fastest programs in the .NET Framework and the JVM client version in the Windows 7 operating system after the use of the A4 variant. The highest acceleration (by 26.52%) was recorded after the application of the A4 variant in the JVM, server version in the Fedora 16 operating system. The uses of variant A2 allow to obtain better acceleration than the use of the A3 variant. The combination of the A2 and A3 variants, the use of the A4 variant, in most cases, allow for even greater accelerations than the use of separate optimizations of variant A2 and A3. However, there are exceptions to this rule, such as in Mono in Windows 7 and in the JVM client version in Fedora 16, where the best was the A2 variant. A surprise is the case of .NET Framework 4.0, where the optimizations applied individually slow down the program, while their combination allows its acceleration. Another curiosity is the .NET Framework 2.0 which behaves differently than the .NET Framework 4.0, since the acceleration occurs only in the A2 variant.

The acceleration in test environment E32

The acceleration in test environment E32 obtained in the three variants listed in the test (A2, A3 and A4), compared to the variant A1, which is not optimized. After applying the A2 and A3 variants in the .NET Framework 4.0, the program has slowed down. The difference in acceleration is not recorded in the JVM server version in the A3 variant in Fedora 11. In other run time environments, the use of optimization has shown acceleration. The largest acceleration (by 24.08%) was recorded in Mono in the Fedora 11 operating system after the application of the A2 variant. The use of variant A2 compared to A3 variant usually allowed to get more acceleration, but there are exceptions (the JVM client version on both operating systems), where it turned out that the A3 variant is better. Generally, the highest acceleration was obtained by the use of the A4 variant, except for Mono in the two operating systems, where the A2 variant happened to be the best.
Again, a surprise is the case of the .NET Framework 4.0, where the optimizations applied individually slow down the program, and their combination allows its acceleration. Similarly, in the environment E64, interestingly the .NET Framework 2.0 behaves differently than the .NET Framework 4.0, because the acceleration has only the A2 variant.

During the code optimization in the A3 variant, one variable result (integer) was removed from the comparison function. The variable described in the A1 variant, where it occurs, should affect the memory consumption at the run-time. Each call to the function should create it again, and the function is called repeatedly in a loop. In order to test, the comparison function in the A1 variant was looped in definitely and memory usage was tested. The tests show that in all run time environments (.NET, Mono, JVM) and in all test environments (E64, E32) memory usage was at a constant level. It follows that the Garbage Collector and the optimizations implemented in all environments can easily handle memory management in this case. In other variants, there was no danger of excessive memory consumption, but they also have been tested and in all variants memory usage was at a constant level.

Below are presented the Java byte codes of comparison function after disassembling in variants A1 (Table 8) and A3 (Table 9). At the first glance, it can be observed that the code in the A3 variant contains fewer instructions that the processor should do. In the JVM runtime environment it corresponds to the greater acceleration. The exception is the JVM server version in the E32 environment in Fedora 11, where there has been no acceleration and the JVM client version in E64 environment in Fedora 16, where a slowdown of the program was observed. Substantial differences in the code is that in the A1 variant three registers are used (iload_1, iload_2, iload_3, istore_3), where as in the variant A3 – only two. The third register is used to store additional variable result. While in the A3 variant, the value of this variable is simply used as a constant (icnst_0 in line 14). However, as shown above, the use of an additional register in the A1 variant does not affect the additional memory consumption. This is due to Garbage Collector of runtime environment which is responsible for memory management. The difference is in the approach to the exit from the function. In the A1 variant instructions are used to move to another place in the code (to the end of the function – ireturn in line 20). Mentioned instructions are go to 19 and if_icmple 19 (go if less or equal). While in the variant A3 there are several exit points from the function (ireturn in lines 6, 13, 15).

Below are presented managed codes, compiled using the csc compiler provided with .NET Framework disassembled by the Ildasm tool. It shows the managed code of comparison function in the A1 variant (Table 10) and A3 (Table 11). It is worth noting that the instructions after compiling with the mcs compiler provided within the Mono environment are identical after disassemblation by the Ildasm tool. At the first glance, it can be observed that the code in the variant A3 contains fewer instructions that the processor should do. But not in all cases that correspond to greater acceleration (e.g. NET Framework 4.0 in all test environments). As in the case of the Java byte
code, in the A1 variant are used three registers (ldarg.1, ldarg.2, stloc.0). The third register is used to store additional variable result. While in the A3 variant, the value of this variable is simply used as a constant (ldc.i4.0 – load constant integer 4 bytes, value 0). However, as shown above (Figure 5), the use of an additional register in the A1 variant does not affect the additional memory consumption, because of Garbage Collector. The difference lies also in the way of exiting from the function: in the A1 variant there is only one exit instruction (ret), while in the A3 variant there are three exit instructions. In the variant A1, in order to achieve identical effect, there are used jump instructions (br.s IL_0010 – jump instruction, ble.s IL_0010 – jump instruction if less or equal), which are aimed at jumping to the instruction of exit from the function.

Tab. 10. Disassembly of the managed code compiled with the csc compiler and used Ildasm tool – comparison function in the A1 variant

```csharp
.method public hidebysig instance int32 compare(int32 x, int32 y) cil managed
{  // Code size 18 (0x12)
    .maxstack 2
    .locals init (int32 V_0)
    IL_0000:  ldc.i4.0
    IL_0001:  stloc.0
    IL_0002:  ldarg.1
    IL_0003:  ldarg.2
    IL_0004:  bge.s IL_0006
    IL_0005:  ret
    IL_0006:  ldc.i4.m1
    IL_0007:  ret
    IL_0008:  ldarg.1
    IL_0009:  ldarg.2
    IL_000a:  bge.s IL_000c
    IL_000b:  ldc.i4.1
    IL_000c:  ret
    IL_000d:  ret
)
```

Reassuming, the best solution in most cases is the use of the early exit from a function and inline expansion. The exception is the Mono runtime environment, where in three on four cases, a better solution is to use only the inline expansion optimization. Second exception is .NET Framework 2.0, where also a better solution is to use only inline expansion. The program in any variant does not show more or less memory consumption. The use of both of this optimizations is also reflected in amore concise code.

7. Summary

The authorial contribution is here the development of the method of early exit from a function and the analysis of acceleration in the component environments after using various optimization. The method of the early exit from a function in certain runtime environments can speed up the program, and in combination with the inline expansion optimization allows speeding up the program, even in cases when the application of such optimizations separately slows the program down. A detailed description and analysis of the test are presented in chapter 6. Performance of the early exit from a function. Benchmarks presented in this paper may help developers to write their own code. On this basis, they can determine that in their case, the optimization would be effective.
Testing is a time-consuming task, so they will not have to undergo the same testing process as shown in article, thanks to the results and conclusions to every test presented here. In the test, also the factors on which effectiveness of optimization method may depend have been pointed. In this article, the intermediate code was analyzed by using reverse engineering methods. The analysis of the intermediate code which provides instructions similar to those that would be executed by the processor, allows to understand the essence of the optimizations.

8. Bibliography

[18] G. Noriskin (Microsoft CLR Performance Team), “Writing High-Performance
Managed Applications: A Primer”, *MSDN Library*, June 2003.


Wcześniejsze wychodzenie z funkcji jako metoda optymalizacji kodu w środowiskach komponentowych

M. KNIOTEK

Artykuł prezentuje użycie metody optymalizacji kodu w środowiskach komponentowych (Java VM, MS .NET, Mono). W zaproponowanej metodzie w środowiskach komponentowych występują przypadki, gdy ręczne zastosowanie optymalizacji pozwala przyspieszyć program. Ze względu na używanie kompilatora JIT (Just In Time) w środowiskach komponentowych, kod nie może być w pełni optymalizowany (ograniczenia czasowe). Kompilator JIT stosuje optymalizacje podczas uruchamiania aktualnie używanej części kodu. Z tego powodu czas użyty na poszukiwanie optymalizacji musi być zrównoważony między czasem oczekiwania przez użytkownika a wyborem zadowalającej optymalizacji. Artykuł prezentuje analizę metody optymalizacji zakończoną wnioskami, które pozwalają odpowiedzieć na pytania: kiedy, czemu oraz w których środowiskach komponentowych należy użyć metody optymalizacji. Prezentowana w artykule autorska metoda nosi nazwę „wcześniejszego wychodzenia z funkcji”.

Słowa kluczowe: optymalizacja kodu, środowiska komponentowe, metoda optymalizacji.
Multi-criteria models of player’s preferences in investment process

P. MICHAŁOWSKI
pmichalowski@wat.edu.pl

Faculty of Cybernetics, Military University of Technology
Kaliskiego Str. 2, 00-908 Warsaw

The paper presents model of stock market players’ preferences in investing process, based on different conditions like the method for valuation financial instruments, player’s character traits including tendency to risk and investing strategies. Author also defined multi-criteria optimization amongst all available decisions, one or many the most important for the player.

Keywords: stock market, behavior models, multi-criteria optimization.

1. Introduction

Decision making in general is based on choice from many possible options – the best for decision maker. In most cases the decision maker wants to reach many targets in the same time. However we cannot assume that there is one general optimal solution. Usually there is a set of many solutions. There are many methods for setting multi-criteria solutions, although each of them needs decision maker’s preferences – information about criteria significance that is subjective and typical for decision maker.

Subjectivism is inevitable here. That is why from mathematical point of view it is important to help the decision maker to make a decision from better perspective but not necessarily the best from other person’s perspective.

Because decision making criteria might be completely different and in the same time incomparable, this is reasonable and in many cases the only one option to find optimal compromise.

While building the optimization problem, we understand it as a task to choose extreme elements from one set according to domination relationship. It defines decision maker’s preferences.

The paper will present the concept of multi-criteria model of stock market player’s preferences in investing process. The article is followed by [1] in terms of building a game theory model of stock market players’ preferences.

2. Decision making method

Let’s assume that players make their decisions based on available to them information. Information available to player n, we will mark as \(I_n\). It is worth to notice that it is built from subsets similar to particular information categories, depending on a used model or models of stock market’s data analysis presented in chapter 4. Let’s assume that every method of financial instruments' valuation is defined by separate subset of information.

Another assumptions that need to be considered are acceptable player’s decisions. According to description in [1] they can decide about issuing a sale or purchase order for chosen stocks or bonds with exact price limits (maximum or minimum). That is why players make sale or purchase decisions based on actual value of financial instruments as well as their forecasted value in the future (time perspective depends on a player). While actual value is known, the forecasted value is estimated by players based on their knowledge (information) and analysis method. Next, from possible decisions’ set, players choose the best ones according to their opinion.

3. Financial instruments valuation depending on chosen method of valuation financial instruments

Several methods of analyzing financial instruments’ valuation have been presented in [2], but the author in this paper focuses on the statement that the player using available information and chosen method, decides to make financial instruments’ sale or purchase decision. Those decisions are based on differences in actual and forecasted prices (in different time perspectives) but also they depend on purchase price. Additionally, for every decision the risk
needs to be estimated. Main valuation criteria will be completeness, (value) information.

General model of player’s price forecasts’ set can be presented as below:

\[ P_F = \{ (P_{T1}, P_{T2}, ..., P_{Tk}) : T = (t_0, t_1, ..., t_n), k, n \in N \} \]  

where:
\( P_F \) – financial instruments’ price forecast’s set \( k \),
\( P_{Tk} \) – forecasting the price of financial instrument \( k \) in the moments of set \( T \),
\( T \) – time perspectives’ set, to simplify we can assume three perspectives: short, medium and long.

A function that generates a set \( P_F \) can be presented like:

\[ f(U, P, I, M) = P_F, \]  

where:
\( A \) – set of financial instruments that player is interested in or that player owns,
\( P \) – set of financial instruments’ actual prices,
\( I \) – set of player’s information,
\( M \) – set of valuation methods used by the player.

At this stage of consideration, key is a form of information that player can access and uses as well as how he/she estimates a risk in function \( f \).

In general way, information is set of data that applies to forecasts of financial instruments price. It informs about possibility of that price possible change, based on different sources. We can assume that information may have form that can be translated to eg. “Company XYZ stocks price will fall down to Q” or, in more general way, “Companies from ABC industry prices will rise by X-Y percent”, which is translated to equation as number (percentage) of potential fall/rise of selected financial instrument price.

One of the most popular methods of analyzing share’s value is **technical analysis**. In its case we can mark player’s information as \( I_T \) (which is a subset of information \( I \)). Technical information, for every financial instrument includes data about e.g.:

- Trend for each financial instrument – trend can be growing, declining or horizontal (no trend)
- Financial instrument’s price over last \( s \) sessions – opening price, closing price, minimum price, maximum price for listing \( s \)
- Technical markers, listed below.
  - Relative Strength Index – presents the strength and maturity of trend. It assumes values from 0 to 100, accurate level is 30 and 70 as levels of potential sale or purchase of the instrument. Extreme values (close to 0 and 100) inform about: reversing the trend for growing or declining.
  \[ RSI = 100 \frac{100}{1 + \frac{C_0 - C_{-n}}{C_{max} - C_{min}}}, \]  
  where:
  \( C_0 \) – actual price,
  \( C_{-n} \) – price \( n \) sessions ago.
  - Stochastic oscillator (\( K%D \)) – based on an assumption that during a growing trend, prices oscillate around higher level of fluctuation range, but in the declining trend – around lower level. Sale signal appears when line \( K\% \) crosses a line \( D\% \) above level of 80. The purchase signal appears when line \( K\% \) crosses line \( D\% \) below level of 20.
  \[ K\% = \frac{C_L - C_{min}}{C_{max} - C_{min}} \times 100, \]  
  where:
  \( C_L \) – last closing price,
  \( C_{min} \) – minimum price from chosen time period,
  \( C_{max} \) – maximum price from chosen time period.
  \( D\% \) is calculated as an arithmetic mean of \( K\% \) value from three last sessions.
  - ATR (Average True Range) – defines shares’ volatility. It assumes that huge volatility is connected to reversal of trends.
  \[ TR = \sup | |C_Y - C_{max}|, |C_Y - C_{min}|, |C_{max} - C_{min-1}| \]  
  where:
  \( C_Y \) – value from yesterday’s closing,
  \( C_{max} \) – today’s maximum value,
  \( C_{min} \) – today’s minimum value,
  \( C_{min-1} \) – yesterday’s minimum value.
  ATR is calculated as a moving average from TR, usually 14-session.
  - ADX (Average Directional Index) – defines actual existing of a trend or informs that what looks as a trend is only a stage of consolidation.
  - OBV (On Balance Volume) – used according to the statement that the trend is strong, if the volume moves according to the prices, is being calculated depending on
prices growth \((OBV = OBV_{-1} + V)\), decrease \((OBV = OBV_{-1} - V)\) or have not changed at all \((OBV = OBV_{-1})\), where:

\[
OBV - OBV \text{ of the current session},
\]

\[
OBV_{-1} - OBV \text{ of the previous session},
\]

\[
V \text{ – circulation on the current session.}
\]

Each marker is described with the value calculated for the exact moment \(t\) and with the “shares” values.

Estimated value of financial instrument is closed in four steps, using information \(I_t\):

- Describing the trend
- Describing the strength of the trend based on markers’ data
- Describing the possible landmarks in trend based on markers’ data
- Describing the financial instrument’s price in time perspective \(T\) (short and medium) based on above data and historical information about the price of financial instrument.

Share’s price is presented below for financial instrument \(k\) in time perspective \(t_1\):

\[
p_{t_1,k} = p_0 + \Delta p_{Tnk} \cdot \partial,
\]

where:

\(p_0\) – current price,

\(\Delta p_{Tnk}\) – change of the price in time range \(T\) where trend accurate actual trend,

\(\partial\) – strength of the trend.

Second most common method of shares’ analyzing used by individual investors is **fundamental analysis**. In this case the player’s information will be marked as \(I_F\), as a subset of information \(I\).

Fundamental analysis is built from five elements:

- Macroeconomic analysis
- Sector analysis
- Situation analysis
- Financial analysis
- Share’s value estimation.

As for the previous assumptions first four steps are there for estimating the risk of actual forecast, and the last step (share’s value estimation) is created to calculate this forecast in time perspective (medium and long).

During next steps of the analysis, the player uses a set of markers typical for each stage and marks their status.

It can be said that for the sake of analysis set of markers are being created for every stage. Their values are described by the player based on information \(I_F\), and next are being summed and averaged which helps to estimate a risk of each forecast. It can be calculated as below:

\[
I_F = (A_M, A_S, A_T, A_F),
\]

where \(A_M, A_S, A_T, A_F\) are the set of information (factors) for these steps.

\[
A_M = (a_{M1}, a_{M2}, ..., a_{MN}), n \in N, a \in C
\]

\[
A_S = (a_{S1}, a_{S2}, ..., a_{SN}), n \in N, a \in C
\]

\[
A_T = (a_{T1}, a_{T2}, ..., a_{TN}), n \in N, a \in C
\]

\[
A_F = (a_{F1}, a_{F2}, ..., a_{FN}), n \in N, a \in C
\]

When it comes to describing the price, fundamental analysis gives many approaches to this issue. In this paper, author presented two most popular:

- Discounted cash flow model
- Solid dividend growth model (Gordon model).

Both of those models are parts of the method of discounted cash flow where company’s assets are overlooked in favor of achieved solid cash flows that the company is or will be able to generate. In that case it is necessary to use a discount which helps to determine updated cash flows in the future.

Calculating the share’s price in discounted cash flow model is presented below for financial instrument \(k\):

\[
p_k = \frac{FCF_0(1+g)}{r-g},
\]

where:

\(FCF_0\) – solid cash flow generated in current year

\(g\) – dividend growth rate,

\(r\) – required return rate determined individual by investor,

It is assumed that \(r > g\).

Gordon model is an example of valuation of the company’s shares based on the cash flows generated for shareholders for which dividends should be paid. Constant dividend growth rate version assumes a constant value of dividend payments. In this case the shares’ valuation needs to be calculated perpetuity (rent’s base – average dividend’s value) discounted by investor’s return rate (cost of the capital).

Calculating the share’s price with this method in presented below for financial instrument \(k\):

\[
p_k = \frac{D_1}{r-g} = \frac{D_0(1+g)}{r-g},
\]
where:
\( g \) – dividend growth rate,
\( D_1 \) – dividend (for one share) paid in next year,
\( D_0 \) – lately paid dividend (for one share),
\( r \) – required return rate, defined individually by the investor.
It is assumed that \( r > g \).

Because investor might use more than one model, we can assume that finally calculated by him price is an average from all results that can be calculated as below:

\[
p_k = \frac{\sum P_{kn}}{n}, \quad (14)
\]

where:
\( P_{kn} \) – financial instrument \( k \) price calculated with model \( n \),
\( n \) – number of used models.

Another methods for valuating financial instruments are for example wallet share analysis or econometric analysis but they are used very rare or not at all by individual investors because of their complexity. More interesting case are players that play without model and base their actions on press releases, brokerage’s recommendations and other sources of information. Those players do not analyze the price of shares, they analyze only information they get.

Because of that we have a situation when price forecast is given by the source. It might be many sources or only one. It is described by below formula:

\[
p_k = \frac{\sum \sigma_n P_{kn}}{n}, \quad (15)
\]

where:
\( \sigma_n \) – investor’s trust for forecast source \( n \), shown in percentage scale 0–100%,
\( P_{kn} \) – price of the instrument \( k \) delivered by source \( n \).

Obviously there is no obstacle for player to use more than one method to valuate financial instruments. Mixing the methods is well known tactic that is helpful in more accurate estimating both value and the risk of price forecast. Very often those methods do not overlap because of different time perspectives.

There is a general formula for financial instrument \( k \) price in time perspective \( t \):

\[
p_{kt} = \frac{\sum \omega_n P_{kn}}{\sum \sigma_n}, \quad (16)
\]

where:
\( p_{kt} \) – price of instrument \( k \) in time perspective \( t(t \in T) \),
\( \omega_n \) – player’s “trust” in method \( n \) measured in percentage scale 0–100%,
\( p_{kn} \) – price of instrument \( k \) in time perspective \( t(t \in T) \), according to method \( n \).

In terms of bonds, their valuation and estimating the investing risk are rather simple because of their structure.

Bonds’ value is defined as a current cash flow value, might be calculated as below for financial instrument \( k \):

\[
p_k = \sum_{t=1}^{n} C_t \left(1+\frac{s}{100}\right)^{-t}, \quad (17)
\]

where:
\( n \) – number of cash flows,
\( C_t \) – cash flow in time \( t \),
\( s \) – required yield.

This is a return rate comparable to an investment in a given bond, depending on the level of interest rates in the financial market (it depends primarily on the expected inflation rate) and risk of default by the issuer of the bonds.

4. Estimating the risk of the price forecast of financial instrument

Every investment is subject to risk. We can name basic risk types as:

- Market risk (connected with investments papers price change, exchange rate change, interest rates change or goods prices changes),
- Credit risk,
- Liquidity risk,
- Operational risk,
- Legal risk.

In this paper our main area of interest is market risk.

One of the most popular methods of risk valuation is Value at Risk. It’s conception assumes that level of losses from investment shouldn’t be bigger than some specified level with assumed probability of players’ forecast mistake.

Let’s assume that player current investment position, which we understand as result of decisions he made to fulfill his objectives is random variable \( X \), where \( \mathcal{X} \subset \mathbb{R} \).

Value At Risk value on \( \alpha \) level can be written as:

\[
VaR_{\alpha}(X) = -\sup\{x \in \mathbb{R} : P(X < x) \leq \alpha\} \quad (18)
\]
Vale at Risk is measurement of maximum loss on whole players’ wallet at assumed $1 - \alpha$ level, which is defined by player himself. For example: private banks set this value to 95%, but KNF (Polish Financial Supervision) prefers 99%.

One problem with Value at Risk is that it haven’t subaddative properties and for different values of $\alpha$ can provide conflicting results of risk valuation. It also doesn’t takes into account loss levels that are higher that set one. But for wide set of elliptical distribution characterized $X$ used in finance Value at Risk is subaddative which was proved in [3].

Three main methods of calculating Value at Risk are:

- Analytical – best when risk factors have normal distribution, but every derogation from theoretical model can lead to error.
- Historical approximation – easier to implement with finding empirical distribution and calculating Value at Risk from it. Main advantage is that this method allows to avoid potential problems with distribution modeling. Main defect is that extreme elements can affect the result.
- Monte Carlo simulations based on hypothetical wallet value $X$, depending on market factors. This method can be used with incomplete data and complicated return functions.

It is worth to calculate Value at Risk with different methods and compare results.

To determinate Value at Risk for wallet return we assume that we set of financial instruments $A$ with prices $P$ (both are $n$-elements). Then in time $t$ vector of prices logarithms $R_t = (lnP_t^1, ..., lnP^n_t)$ has non-generated, $n$-dimensional normal distribution with expected value vector $R_0 + \mathbf{\bar{R}}t$ and co-variation matrix $t \Sigma$.

Assuming that current player wallet contains $m$ instruments, we can present it’s current prices, which depends on basic instruments prices, as: $U_1 = U_1(R_t), ..., U_m = = U_m(R_t)$.

Wallet value at moment $t$ can be presented as:

$$V(R_t) = a + \sum_{i=1}^{n} \Delta_i R_t^i = a + \Delta^T R_t \quad (20)$$

where $a \in R, \Delta \in R^n$.

Let’s assume now some time horizon $T$, for which we want to estimate Value at Risk for players’ wallet. $VaR(\alpha)$ will be Value at Risk on $\alpha$ level for wallet returns, which mean position:

$$X = V_T - V_0 = V(R_T) - V(R_0). \quad (21)$$

We know from [3] that $VaR(\alpha)$ satisfies:

$$P(V(R_T) - V(R_0) \leq - VaR(\alpha)) = \alpha \quad (22)$$

So Value at Risk for linear wallet, for $T$ time horizon on level $\alpha$ is:

$$VaR(\alpha) = q_{1-\alpha} \sqrt{\Delta T (\Sigma T) \Delta - \Delta^T \mathbf{\bar{R}} T} \quad (23)$$

where $q_\alpha$ is $\alpha$-level quantile with $N(0,1)$ distribution.

But for purposes of player strategy definition, he can just take into account only his current wallet or his wallet and in addition instruments he’s interested in. So we can assume that player will calculate Value at Risk for every element of set $V$, which contains all combinations without repetitions of instruments in set $A$, so he can analyze every combination of his current assets and assets which purchase he’s considering.

5. The model of players’ preference in terms of character traits

As shown in the previous section, each forecast value of the instrument is subject to the estimated level of risk.

Risk aversion determines human (in this case – player) behavior, when he is exposed to uncertainty conditions, when he is trying to reduce that uncertainty.

Risk aversion characterizes with ability to accept lower payment but also connected to lower risk, instead of potentially higher but more risky payment.

It can be assumed the three basic type of player, due to their risk approach:

- Risk-averse,
- Risk-neutral ($\alpha$ ability to risk),
- Risk-seeking.

In utility theory there are assumptions that lottery utility is sum of utility of it possible results weighted with probability of their occurrence.
\[ E(u(x)) = \sum p_i u(x_i) \]  
(24)

Decision-maker in assumption is trying to maximize his expected utility.

We can say that player is risk-neutral when there is indifference for him between taking part in lottery and getting certain reward.

\[ E(u(x)) = u(E(x)) \]  
(25)

Risk-seeking player submit taking part in lottery than getting certain reward.

\[ E(u(x)) > u(E(x)) \]  
(26)

Risk-averse player prefers getting certain reward than taking part in lottery.

\[ E(u(x)) < u(E(x)) \]  
(27)

One of methods of measuring total risk aversion is Arrow-Pratt measure defined as:

\[ R(x) = - \frac{u''(x)}{u'(x)} \]  
(28)

for \( u(x) \) being player utility function, and \( u'(x), u''(x) \) are first and second derivative.

That equation gives positive results for concave \( u''(x) < 0 \) and negative results for convex \( u''(x) > 0 \) utility function \( u(x) \).

The more a player is risk-seeking, the more convex is utility function and \( R(x) \) gives lower values. Analogically, for more risk-averse player utility function is more concave and \( R(x) \) gives higher values.

In newest wallet theories, risk aversion is measured as additional reward (payment) which player’s requires accepting additional risk.

6. The model of players’ preference in terms of used strategies

Strategies used by the players can be very diverse. They often depend on the method of valuation of financial instruments, which the player uses because it provides specific information, and is based on specific assumptions. For example, players can play according to trends against them, play on sudden fluctuations of trends, falls, rises, etc. There are many possibilities.

We can define strategy as set of elements, defined as financial instrument value change, time and decision.

For purposes of this paper let’s assume that player can take one of three decisions: sold, keep, buy.

From the model point of view it is not important, what specific strategy player uses. It is important to assume that any strategy can be characterized by the elements that define it which can be presented as follows:

\[ S_k = \{s_{k1}, s_{k2}, ..., s_{kn}\}, n \in N \]  
(29)

Each player then has a list of strategies he uses, so you can assume that the player while making decision, determines which elements of the strategy have been met and then compares it with the strategies and their components, which he used by himself.

For simplification we can assume that strategies comparison is made by comprising their elements one by one and calculating similar and opposite one of them (eg. by calculating percentage of similarity).

7. The model of preference in investment decisions made by different type of players – multi-criteria optimization

Based on the foregoing, we can determine that the set of ratings, we will describe a player’s decisions:

\[ D = (F, V, R, S) \]  
(30)

where:

\( D \) – player’s decision evaluation,

\( F \) – player’s profit forecast,

\( V \) – player’s risk value,

\( R \) – player’s risk approach,

\( S \) – evaluation of the strategy regarding the decision with respect to the player’s strategy.

Players based on these four values describing a single decision, tries to find the best (optimal) elements.

We assume that a player is trying to maximize the expected return on investment, minimize risk fitting his risk approach and maximize fitting the strategy which can be accurate for a several investment that fits into set of its strategies.

It can be presented as below:

\[ f^* \in F, F_1(f^*) = \sup_{f \in F} F_1(f) \]  
(31)

\[ v^* \in R, F_2(v^*) = \sup_{v \in R} F_2(v) \]  
(32)

\[ s^* \in S, F_3(s^*) = \sup_{s \in S} F_3(s) \]  
(33)
where:

$F_1, F_2, F_3$ – functions defining value of factors.

Let’s assume that profit maximization is also understood as the minimization of losses, which means that the player should want to get rid of financial instruments that could bring him the greatest disadvantage. Relying on maximizing the return rate will cause (in case of loss) total exclusion the instrument from a set of best shares, as in this case, the return is negative. We should, therefore, rely on the absolute value of the return rate:

$$ f^* \in F, F_1(|f^*|) = \sup_{f \in F} F_1(f). \quad (34) $$

We can see that we have a responsibility to maximize the two quality indicators in three-dimensional space $F_1(k) = y_1, F_2(k) = y_2$ and minimizing third indicator of value: $F_3(k) = y_3$. From the point of view of only the first indicator $F$ would be ideal to achieve value $y_1^{max}$. From the point of view of only the second indicator $S$ ideally be as high as $y_2^{max}$. However, from the point of view of the third ratio $R$ would reach a value which is ideal $y_3^{min}$. Assuming, therefore, that the decision-maker seeks to maximize the simultaneous minimization of the two indices and the third, we can say that the purpose of selecting the solution reaches $d^*$.

$$ F(d^*) = y^* = (y_1^{max}, y_2^{max}, y_3^{min}). \quad (35) $$

That kind of solution usually is not in set of acceptable solutions. It is expressed that image $y^*$ of that solutions lies outside set $D$ of ratings of acceptable solutions. Point $y^*$, which coordinates are the result of adopted preferences model “≥” may be considered as player goal. This is the point to which player wants to get as close as possible.

In situation, when $y^* \notin D$, we usually formulate additional optimization problem, that allow us to set acceptable solution, laying closest to the target $y^*$.

We can assume that Euclidean distance of point representing $k$ decision from target (best point, marked as $K$) is quality measure of that decision. Propositional dominant $\varphi(k_1, k_2)$ can be presented as below:

$$ \varphi(k_1, k_2) = \|K, k_1\| \geq \|K, k_2\| \quad (36) $$

Using the function $\varphi$ we can describe a domination relation $R_\varphi$

$$ R_\varphi = \{(y, z) \in B \times B | \varphi(y, z)\} \quad (37) $$

The set of possible decisions is primarily limited by three factors:

- Player’s risk,
- Minimum sale price,
- Maximum purchase price.

$$ v_k \leq v_{max}, \forall r_k \geq f_{min} \quad (38) $$

$$ f_{k} \geq f_{min}, \forall buying \quad (39) $$

$$ f_{k} \leq f_{max}, \forall selling. \quad (40) $$

As optimal element (best in recognition of the decision maker) will be called one which is better in terms of relations of domination $R_\varphi$ than all the other elements of the set $D$. It can be described as: $(y^*, y) \in R, \sum y \in D \{|y^*\}$. Element $y^*$ is dominated, and set of all elements is described as:

$$ D_B^R = \{y^* \in D | (y^*, y) \in R, \sum y \in D \{|y^*\} \quad (41) $$

In case when set $D_B^R$ was empty, which happens often, decision maker must be pleased with the element $y^* \in D$, which is the best in set $D$. This element is called not dominated in relation $R$, and set of all dominated elements is described:

$$ D_N^R = \{y^* \in D | \{y \in Y, y \neq y^*} = \emptyset, (y, y^*) \in R \} \quad (42) $$

The dominant and not dominants are known as “elements of the optimal” (extreme elements). Thus, the solution of the optimization problem $(D, R)$ are elements of sets $D_B^R$, $D_N^R$.

In situation where both $D_B^R$, and $D_N^R$ are empty sets, you can use compromise solutions or sub solutions.

Since the player does not have to choose only one decision at a time, and can make them a lot in the same time, the player with the harvest and dominant components chooses another solution ranking them using previously mentioned solutions (ideal).

8. Summary

Presented model of behavior, and precisely optimal decision making model the best for player is another step in building complete model of mass stock investors’ behavior in investment process.

Preferences model has been defined generally in order to present a general way of decision making by investors. Another step should be defining precise model for chosen method.
The most important is to compile the exact algorithms for building the price forecasts, and next attempt to simulate players’ behavior in closed set of players on limited number of financial instruments, based only on actual information about their value, brokerage reports as well as historical data.

The next step might be attempt to study the statistics of valuation methods used by individual players and defining their characterological structure (e.g. ability to risk).

Another issue that needs to be considered is setting the game solution – its state of balance, if such exists.

9. Bibliography


[27] Szczegółowe zasady obrotu giełdowego, tekst ujednolicony według stanu prawnego na dzień 1 maja 2012 r., http://www.gpw.pl
Wielokryterialne modele preferencji graczy giełdowych w procesie inwestowania

P. MICHAŁOWSKI

W artykule przedstawiono model preferencji graczy giełdowych w procesie inwestowania, opierając się na zróżnicowanych przesłankach, jak użycia metoda wyceny instrumentów finansowych, cechy charakterologiczne gracza, w tym skłonność do ryzyka, oraz stosowane strategie inwestowania. Następnie zdefiniowano zadanie optymalizacji wielokryterialnej wyboru spośród wszystkich dostępnych decyzji jednej lub wielu, w rozumieniu gracza, najlepszych.

Słowa kluczowe: giełda, modele zachowań, optymalizacja wielokryterialna.
A multifaceted model for software reliability prediction during testing

R. PEŁKA
radoslaw.pelka@wat.edu.pl

Institute of Computer and Information Systems
Faculty of Cybernetics, Military University of Technology
Kaliskiego Str. 2, 00-908 Warsaw, Poland

Analysis of software reliability plays an important role in quality assurance plan realization during software development. By monitoring changes of evaluated reliability in relation to quality objectives it is possible to analyze current situation in respect to agreed requirements and initiate appropriate actions when needed to secure fulfilling of the goals. The use of software reliability growth models as the only method for reliability evaluation seems to be too much simplified approach. Such approach, based solely on fault detection history, may in some circumstances be risky and lead to significantly wrong decisions related to the software validation process. Taking possible pros and cons into account the model described in this paper is proposed to use a number of additional information concerning the software being tested and the validation process itself, to produce more accurate outcomes from the reliability analysis. The produced outcome gives an appropriate feedback for a decision makers, taking into account assumed software development process characteristic. Integral part of the presented approach is devoted to reliability characteristics of a system being tested in parallel by several independent teams.

Keywords: reliability, software, modeling, testing.

1. Introduction

Software reliability analysis plays an important role in overall quality assurance plan in software development process. Nowadays, when business realities force companies to be more competitive when it comes to faster deliveries of a new software to the market, software reliability remains a crucial factor which determines the final success of a product. Such situation provoke existence of a development process optimization procedures that incorporate reliability objectives as a main criteria. Such approach may be potentially good but on the other hand it may also be dangerous when reliability findings are not proper. To make sure that produced reliability evaluations can be rely on, it is crucial to secure that the applied methods allow to incorporate possibly large range of important aspects related to software verification process and system under testing as such. This kind of approach is more demanding for persons performing quality analysis because it entails a need to collect, prepare and process larger amount of data, compared to situation when only software reliability growth models (SRGMs) are used [1]. Successful application of SRGMs has been proved many times [2]. Therefore, the approach proposed in this paper incorporates use of an SRGM as a method for software reliability prediction, however, taking all the related disadvantages into account [3], [4], additional extensions have been proposed. The extensions are used to make it possible, in a given moment of software validation process, to produce more accurate outcomes for a decisive persons, based on results provided by SRGM but processed in accordance with taken assumptions. The main goal for this version of the model was to verify methods of information synthesis and influence of such approach on the quality and financial results of a software development project execution when software validation is supposed to be conducted under strictly defined reliability objectives.

The subject of interest and research area for the proposed approach are large systems consisting of many functional modules, implementing logic for various and complex tasks. In case of such systems it is a common situation that implementation of functional extension to the software is performed simultaneously by several development teams. To a large extend the teams may work independently of each other, practically up to the final integration phase when effects of their work are joined with system being prepared for a customer. From software reliability analysis point of view each such individual process being realized by a single team is important. As well, important is a main process which consists of all the individual processes.
Taking into account the specificity of software development process conducted in the mentioned way, appropriate methods for an individual process are described in chapters 2–6, and methods for a main process are described in chapter 7. Chapter 8 contains summary of the results of application of the model in a real software development project.

2. Software reliability evaluation based on faults detection history

A starting point in the proposed model is to evaluate software reliability, based on information concerning history of faults detection during system testing. For this purpose the Musa and Okumoto software reliability growth model [5], based on non-homogeneous Poisson process theory, was used. This model was selected due to proven effectiveness in practical application and satisfactory level of fit of the model to data representing history of defects detection in the examined software. In order to evaluate the fit of model to available data, three criteria which are widely used for the purpose of SRGMs comparative analysis [6] can be applied. The criteria are the mean squared errors (1), the predictive-ratio risk (2), and Akaike’s information criterion (3).

\[
MSE = \frac{\sum_{i=1}^{u} [m(t_i) - y_i]^2}{u-N}, \quad (1)
\]

where:
\(y_i\) – number of faults detected until time \(t_i\);
\(t_i\) – time of \(i\)-th fault detection;
\(u\) – number of data concerning fault detection times, \(u > N\);
\(N\) – number of model parameters;
\(m(t_i)\) – expected cumulative number of faults at time \(t_i, i = 1,u\).

\[
P RR = \sum_{i=1}^{u} \left( \frac{m(t_i) - y_i}{m(t_i)} \right)^2, \quad (2)
\]

where:
\(y_i\) – number of faults detected until time \(t_i\);
\(t_i\) – time of \(i\)-th fault detection;
\(u\) – number of data concerning fault detection times;
\(m(t_i)\) – expected cumulative number of faults at time \(t_i, i = 1,u\).

\[
AIC = 2q - 2 \ln(L), \quad (3)
\]

where:
\(q\) – number of model parameters;
\(L\) – maximum likelihood function of the selected model.

The Musa-Okumoto model belongs to the class of models with infinite number of faults. Due to the form of the mean value function (4) this model is classified as logarithmic model. In such case intensity of failures decreases exponentially, along with detection of subsequent faults. Therefore, tendency to detect more faults in the early phase of testing is incorporated in the model.

\[
m(t) = \beta_0 \ln(\beta_1 t + 1), \quad (4)
\]

where:
\(\beta_0 = \frac{1}{\theta}\)
\(\beta_1 = \lambda_0 \theta\)
\(\lambda_0\) – initial failure intensity;
\(\theta\) – rate of reduction in the normalized failure intensity per failure, \(\theta > 0\).

For the Musa-Okumoto model appropriate form of the maximum likelihood function (5), required for Akaike’s criterion calculation, was determined based on general form of Poisson distribution probability density function.

\[
L(\beta_0, \beta_1 | t_1, t_2, ..., t_u) = u \ln \beta_0 + u \ln \beta_1 - \ldots - \beta_0 \ln(\beta_1 t_u + 1) - \sum_{i=1}^{u} \ln(\beta_1 t_i + 1), \quad (5)
\]

where:
\(t_i\) – time of \(i\)-th fault detection;
\(t_u\) – time of \(u\)-th fault detection;
\(u\) – number of data concerning fault detection times.

For a given moment in software validation process appropriate values for \(\beta_0\) and \(\beta_1\) parameters can be determined by estimators based on maximum likelihood method. Estimators obtained by this method are usually characterized by at least consistency, asymptotic normality and asymptotic efficiency. Having the model parameters evaluated it is possible to determine value of conditional reliability function (6) for time period \(t + x\).

\[
R(x|t) = \frac{R(t+x)}{R(t)} = \left( \frac{\beta_0 t + 1}{\beta_1 (t+x) + 1} \right)^{\beta_0}, \quad (6)
\]

where:
\(R(t) = e^{-m(t)}\)
\(m(t)\) is defined by (4).
3. Reliability evaluation risk factor

When software reliability evaluation for a given time horizon is determined, it is then a relevant question how much the evaluation is credible in context of the system under testing as such and current stage of the validation process. To be able to incorporate aspects that have the potential to influence results of the ongoing reliability analysis, a reliability evaluation risk factor is proposed. The risk factor is supposed to be built on information concerning risk of reliability evaluations from a single module perspective, together with information about significance of the module from system perspective (a module weight). The risk factor is supposed to be a function of time, where time is discretized, with step equal \( k \). A step length shall be set to e.g. an hour, a day or a week. A step length shall be set the way that its value corresponds to the characteristic and pace of the validation process realization. It should be relatively shorter than whole planned validation period and relatively longer than execution time of a single test case.

Required information about risk from a single module perspective is built on data concerning test coverage and adequacy of the number of faults detected in a module compare to the expected value. The expected number of faults for a given module is determined based on historical data analysis, taking into account scale of current development project (7). It is assumed that continues development of software from a system module, by using the same programming paradigm in each of the development projects, gives enough argument to perceive the software as to be homogeneous from reliability perspective. In case of lack of information concerning faults detected in a given module, the expected number of faults can be determined by applying method based on a program volume, proposed by Halstead [7].

\[ O_i = \left[ L_i \cdot \frac{\sum_{j=1}^{m} y_{ij}}{\sum_{j=1}^{m} Y_{ij}} \right], \]  

(7)

where:
- \( m \) – number of historical projects, \( m \in \mathbb{N} \);
- \( y_{ij} \) – number of faults in \( i \)-th module from \( j \)-th historical project, \( y_{ij} \in \mathbb{N} \cup \{0\}, \ i = 1, m; \ j = 1, m; \)
- \( L_i \) – number of new or modified lines of code in \( i \)-th module from current project, \( L_i \in \mathbb{N}, \ i = 1, m; \)

\( Y_{ij} \) – number of new or modified lines of code in \( i \)-th module from \( j \)-th historical project, \( Y_{ij} \in \mathbb{N}, \ i = 1, m; \ j = 1, m \).

Equation (8) shows formula of the function used to assess value of adequacy of the number of faults detected in a given module, in a given moment of the validation process (\( k \)), compare to the expected value produced by formula (7). The constant \( a \) is a decisive variable whose value shall express the belief of a decisive person about importance of such a fact that number of faults detected in a module differ from the expected value. Value of constant \( a \) shall basically not exceed value 2. Higher values lead to situation when even small deviation from the expected value causes significant increase of the risk factor value.

\[ A_i(k) = \begin{cases} a d_i(k), & \text{when } d_i(k) < 0 \\ a - d_i(k), & \text{when } d_i(k) \geq 0 \end{cases}, \]  

(8)

where:
- \( d_i(k) = o_i - p_i^{(k)} \)
- \( o_i \) – expected number of faults for the \( i \)-th module, \( i = 1, \bar{m} \);
- \( p_i^{(k)} \) – number of faults detected in the \( i \)-th module until end of \( k \) step, \( i = 1, \bar{m} \);
- \( a \) – a constant influencing the shape of adequcy function, \( a \geq 1 \).

Equation (9) shows formula of the risk factor for a single \( i \)-th module modified in current development project. The formula was constructed the way that it takes into account deviation of the number of faults detected in a given module during software validation, from the number of faults expected for this module. It was assumed that as long as there are still some planned but not executed test cases that covers functionality provided by a given module (note that 0 means 0% coverage while 1 means 100% coverage), value of the risk factor for this module is always higher than zero.

When number of faults detected in a given module differ from the expected value determined based on historical data, the risk factor value is additionally increased, proportionally to the value of the adequacy factor. For modules not modified in current development project, the risk factor is assumed to be equal zero.

\[ h_i(k) = (1 - c_i^{(k)}) A_i(k), \]  

(9)
where:
\( c_i^{(k)} \) – test coverage of the i-th module until end of k step, \( c_i^{(k)} \in [0,1], i = \overline{1,n}; \)
\( n \) – number of modules in the system, \( n \in N \).

To be able to determine necessary values of weights of the modules, all the modules shall be classified based on code complexity and functional criticality analysis. This way, appropriate weight value can be given for particular class of modules. Equation (10) is used to determine a numerical value reflecting importance of i-th module from the system reliability examination point of view.

\[
l_i = Z_i(1,25 - 0,25F_i), \quad (10)
\]

where:
\( Z_i \) – complexity of i-th module, \( Z_i \in [0,1], i = \overline{1,n}; \)
\( F_i \) – functional criticality of i-th module, \( F_i \in \{1,2,3,4\}, i = \overline{1,n}; \)
\( n \) – number of modules in the system, \( n \in N \).

Appropriate values of functional criticality \( (F_i) \) for all modules are determined by experts having extensive knowledge about examined system. The main criterion here is the impact of a failure in a given module on overall ability of the system to perform its tasks. The meaning of particular values used for expression of the functional criticality is as follows:

1 – high importance module;
2 – normal importance module;
3 – low importance module;
4 – auxiliary module.

Complexity of a given module \( (Z_i) \) is determined based on a combination of selected code complexity metrics. The selected metrics shall be appropriate for the type of examined code and shall be characterized by low level of mutual correlation, to maximize effectiveness of their use in the decisive process. In this research the McCabe’s cyclomatic complexity [8] and data flow complexity metrics were used. The latter metric is represented by equation (11).

\[
z_i(PD) = (lid_i + liz_i) \cdot (ldd_i + ldz_i + ldp_i), \quad (11)
\]

where:
\( lid_i \) – number of interfaces incoming to module, \( lid_i \in N \cup \{0\}, i = \overline{1,n}; \)
\( liz_i \) – number of interfaces outgoing from module, \( liz_i \in N \cup \{0\}, i = \overline{1,n}; \)
\( ldd_i \) – number of data incoming to module, \( ldd_i \in N \cup \{0\}, i = \overline{1,n}; \)
\( ldz_i \) – number of data outgoing from module, \( ldz_i \in N \cup \{0\}, i = \overline{1,n}; \)
\( ldp_i \) – number of permanent data maintained by module, \( ldp_i \in N \cup \{0\}, i = \overline{1,n}; \)
\( n \) – number of modules in the system, \( n \in N \).

Each of the used metrics focuses on different aspects of software engineering, thus they characterize complexity of a given module in a different manner. Due to that it is possible to achieve relatively better evaluation of the overall module complexity, compare to situation when used metrics belong to the same class. The overall complexity for a given module is defined as a product of values given by single metrics. Having the overall complexity calculated for all the system modules, all the values are then normalized to range \([0, 1]\).

When values of the importance factor are calculated for each of system module, based on equation (10), the next step is to decide which of the received values are going to be used as thresholds for assigning the modules to different classes. For this reason the received values are first sorted from lowest to highest. Then, it is decided how many percent of modules with the lowest values belongs to the first class and how many modules with the highest values belongs to the third class. Value of the importance factor calculated for a module which is the first one, according to the determined order, that belongs to the second class of modules constitutes the first threshold value. Value of the importance factor calculated for a module which is the last one, according to the determined order, that belongs to the second class of modules constitutes the second threshold value. If by \( \delta_1 \) we denote the first threshold value and by \( \delta_2 \) the second threshold value, then appropriate weight values for modules are determined by the formula (12).

\[
w_i = \begin{cases} 
1 & \text{when } l_i < \delta_1 \\
2 & \text{when } \delta_1 \leq l_i \leq \delta_2 \\
4 & \text{when } l_i > \delta_2 
\end{cases}, \quad (12)
\]

where:
\( i = \overline{1,n} \)
\( n \) – number of module in the system, \( n \in N \).

Form of the reliability evaluation risk factor that is supposed to be used for the entire system is finally defined by formula (13). It is defined as weighted arithmetic mean of the individual risk factors of system modules.
\[ H(k) = \frac{\sum_{i=1}^{n} w_i h_i(k)}{\sum_{i=1}^{n} w_i} \quad (13) \]

The higher is value of the risk factor (13), the higher is risk that reliability predictions for the system under testing are not adequate to its actual reliability. To calculate value of the risk factor, data related to the modules modified in current development project are used. Obtained value is used in further analysis where, together with reliability evaluations produced by selected SRGM, as well as other crucial information concerning the system under testing, it is used to determine value of the maturity and readiness to integration indicator. By integration in this case it is meant that modules modified in a given individual development process are merged with the system being prepared for a customer.

4. Fault density adequacy factor

Software reliability evaluation produced by selected SRGM, together with information provided by the reliability evaluation risk factor, are based on only part of valuable information usually available for a decisive person during software validation process. From the model completeness perspective as well as to improve quality of produced outcomes, especially during early phase of the validation process, it seems to be reasonable to additionally use data related to software development process as such, data being a result of experts opinions and historical data concerning the system under testing. In software development process synthesis of various information that may be valuable from strategic decisions perspective, especially decision about delivery of a final product to a customer, is essential in today’s reality.

Fault density adequacy factor is supposed to reflect the level of fault density reached at a given moment in software validation process, in respect to the fault density requirements defined by a decisive person. Value of the factor is determined based on number of faults already detected in the software at a given moment in time, together with predicted number of faults that have been introduced into the system during code modification. The number of already detected faults is known at a given moment in time. The number of faults introduced into the system is determined by use of a Bayes net [10]. Structure of the used net is presented in figure 1.

Fig. 1. Structure of a Bayes net used to evaluate number of faults introduced into the system

The number of faults introduced into the system is made dependent on selected information concerning the software development process. For a net node which is not descendant of any other node, according to the net structure, appropriate value can be specified by experts working in the development project which is being examined. Values for the remaining nodes are determined based on information provided by nodes which are parent nodes to a given node, according to the net structure. Allowed values for such a node are characterized by suitable probability distribution. Determined this way expected number of faults introduced into the system is used in further analysis.

At a given moment in the software validation process, the current fault density for the software under testing can be evaluated according to formula (14).

\[ G_o = s \cdot \frac{b_o - b_w}{L} \quad (14) \]

where:
- \( s \) – scale factor; represents number of source code lines used as a base for defining fault density requirement;
- \( b_o \) – expected number of faults introduced into the system;
- \( b_w \) – number of faults found until end of a given time step;
- \( L \) – number of new or modified source code lines in current development project.

For the purpose of fault density evaluation an assumption is taken that the software is free of faults at the moment when development of new version of system starts. However, it is possible that during software validation process some legacy, previously unknown faults are revealed. That’s why if number of faults detected until end of a given time step is higher
than the expected number of faults then fault density is assumed to be equal zero.

Having the software fault density evaluated it is then possible to determine value of the fault density adequacy factor $A_g(k)$. As a main reference the value of required fault density is used which, together with tolerance between value required and value achieved, is defined by a decisive person. The fault density adequacy factor takes values from range $[0,1]$, with step which equals 0.1. Value 1 in this case means that achieved fault density is equal or lower than the required one. Based on the defined tolerance the subsequent fault density thresholds are determined. The defined tolerance reflects a ten percent threshold which, when crossed, results in value of the fault density adequacy factor decreased by 0.1, until it reaches 0. If by $\rho$ we denote the required fault density and by $\tau$ the tolerance between value required and value achieved, then appropriate value of the fault density adequacy factor can be determined by the formula (15). Decided this way value of the factor is used later to determine value of the maturity and readiness to integration indicator.

$$A_g(k) = \begin{cases} 
1 & \text{when } G_0 \leq \rho + 0 \cdot \tau \\
0.9 & \text{when } G_0 \leq \rho + 1 \cdot \tau \\
0.8 & \text{when } G_0 \leq \rho + 2 \cdot \tau \\
\vdots & \\
0.2 & \text{when } G_0 \leq \rho + 8 \cdot \tau \\
0.1 & \text{when } G_0 \leq \rho + 9 \cdot \tau \\
0 & \text{when } G_0 > \rho + 9 \cdot \tau 
\end{cases} \quad (15)$$

5. Maturity and readiness to integration indicator

Presented model introduces concept of a maturity and readiness to integration indicator which is supposed to be used as a guidance for a decisive person when decisions about integration of modified software modules with system that is being prepared for a customer are taken. Meaning of the indicator is directly related to development methodology applied for the system under testing. The way value for the indicator is determined is characterized by synthesis of information of different type, to finally provide a single value appropriate for a decisive process. In the presented model the decision that is supposed to be taken during validation process lies in the fact to agree or not agree on integration of modified software modules with system that is being prepared for a customer, based on the defined reliability objectives. By applying the presented approach the risk of taking wrong decision, that is decision to integrate modified modules while the software under testing has not reached appropriate level of its reliability, is reduced. It is assumed that reliability of examined software is a priority criterion for the decisive person.

The maturity and readiness to integration indicator expresses, by a percentage value, the level of fulfilling the requirement for software reliability defined by a decisive person. Value of the indicator for a given moment of software validation process is determined in the following way. The software reliability objective defined by a decisive person constitutes a level which, when reached, means 100% fulfillment of the requirement. First step is to compare software reliability evaluated by the selected SRGM, in case of this research by model Musa-Okumoto described in chapter 2, with level defined as the objective, to get preliminary level of the requirement fulfillment. For instance, when objective is set to 0.8 and evaluated reliability is 0.6 then the preliminary level is 75%. Then, it is assumed that the preliminary level can be treated as the final one when there are no symptoms showing that value of the evaluated software reliability might not be proper. By the symptoms the counted values of the reliability evaluation risk factor (13) and the fault density adequacy factor (15) are meant. It is assumed that when product $(1 - H(k)) \cdot A_g(k)$ equals 1 then the reliability evaluated by the selected SRGM is the final one and so the percentage value related to the evaluated reliability constitutes value of the maturity and readiness to integration indicator. Otherwise, that is when the above product is less than 1, appropriate percentage value to be used as the maturity and readiness to integration indicator value is counted by a proportional reduction in the percentage value corresponding to the evaluated software reliability. For instance, when the product gives 0.4 and previously counted preliminary level is 75% then the maturity and readiness to integration indicator gets value 30%.

6. Cost function

Taking into account economic side of software development project, in particular costs related to realization of software validation process and maintenance of the product on customer side, the proposed model introduces as well
a cost function (16). The function allows to keep track of changes concerning overall costs for the ongoing project, taking into account real data available at a given moment in software validation process as well as predicted data concerning future. The cost aspect introduces additional limitation imposed on the main plan aiming to produce software with satisfying level of reliability but within decided budget. Therefore, together with quality analysis there is also cost analysis being performed for system under testing.

\[ C(x|k) = C_f \cdot N(k) + C_h \cdot k + C_m \cdot (N(k + x) - N(k)), \]  

where:

- \( k \) – current time (corresponds to end of time step \( k \));
- \( x \) – additional time;
- \( N(k) \) – number of faults detected until end of time step \( k \);
- \( N(k + x) \) – predicted number of faults until end of time \( k + x \);
- \( C(x|k) \) – predicted overall cost until end of time \( k + x \);
- \( C_f \) – cost of single fault removal during software validation;
- \( C_h \) – cost of conducting software validation during a single time step \( k \);
- \( C_m \) – cost of single fault removal during software utilization by a customer.

Naturally, cost of fault removal when fault is detected by a customer is much higher than cost that needs to be incurred when fault removal takes place during software validation. It is assumed that cost of conducting the validation process as such is not negligible. By inclusion of this cost into analysis it is possible to judge whether to continue the validation process or not. It might be important especially in case when the quality goals have almost been met while budget limits are already or closely reached. The number of faults detected until end of time step \( k \) is known at a given moment when cost calculation takes place. The number of faults detected until additional time \( x \) passes is predicted by the selected SRGM. In case of this research it is model Musa-Okumoto described in chapter 2.

7. Reliability characteristics for software parallel validation

In case of developing a big scale system it is possible that many small production processes coexist and are realized, to the large extend, independently of each other by many development teams. Form the final product perspective it is however necessary to be able to monitor interesting characteristics for the main production process which aims to create a new version of the system.

Considering validation process which is being realized by a team involved in a single production process, two separable states can be defined, representing situation in which the process at a given moment in time can be. The first used state indicates searching for faults while the second one indicates fixing faults. This fact was used to describe a validation process together with relevant reliability characteristics. The following assumptions were taken:

- times between consecutive faults detection are independent random variables with the same probability distribution,
- faults detection as well as faults fixing intensity remain constant during examined time intervals and are the same for all independent validation processes of independent production processes; the intensity values are initially decided by domain experts, based on development teams characteristics, planned tasks characteristics, as well as development methodology; values of the intensities may be changed due to new evaluations based on fault detection and fault fixing information from the already finished time intervals,
- the number of testing teams is constant, equals the number of independent development processes, and each of the individual validation processes can be in one of two possible states – searching for faults or fixing a fault.

In case of individual validation process a state machine with two states is applied. Analysis of such process can be based on stochastic process theory, in particular theory of Markov process with continuous time and discrete set of states [11]. It was assumed that time of being in particular state of the process is characterized by exponential probability distribution with known value of parameter reflecting intensity with which the process leaves a state. Probability density for variables which represent time of being in fault detection state and time of being in fault fixing state are described by equations (17) and (18) respectively.

\[ u(t) = \lambda e^{-\lambda t}, \]  

where:

- \( \lambda \) – fault detection intensity in individual validation process.
\[ u(t) = \lambda e^{-\lambda t} \]  

(18)

where:
\( \mu \) – fault fixing intensity in individual validation process.

From the main process perspective at a given moment in time some of the running in parallel individual processes are in fault detection state and some are in fault fixing state. Based on that fact it is possible to define a set of states for the main process, where state denoted as \( k \) means that currently \( k \) among all of the individual processes are in fault fixing state, whereas remaining \( n-k \) processes are in fault detection state. A Markov chain state diagram for the main process, which is a time-homogeneous Markov chain with finite state space, is presented in figure 2.

\[ f(t, k) = (k\mu + (n-k)\lambda)e^{-(k\mu+(n-k)\lambda)t} \]  

(19)

where:
\( n \) – total number of individual processes constituting the main process;
\( k \) – number of individual processes being in fault fixing state;
\( (n-k) \) – number of individual processes being in fault detection state;
\( \lambda \) – intensity of fault detection in individual validation process;
\( \mu \) – intensity of fault fixing in individual validation process.

Value of probability of state \( k \) to state \( k-1 \) transition and value of probability of state \( k \) to state \( k+1 \) transition in the main process is determined based on equations (20) and (21) respectively.

\[ q(k-1|k) = \frac{k\mu}{k\mu+(n-k)\lambda} \]  

(20)

\[ q(k+1|k) = \frac{(n-k)\lambda}{k\mu+(n-k)\lambda} \]  

(21)

An unconditional probability of being in a given state of the main process is determined based on equation (22) which is a result of transformation of the Markov chain balance equation.

\[ p(k) = \frac{\binom{n}{k} (\frac{\lambda}{\mu})^k}{(1+\frac{\lambda}{\mu})^n} \]  

(22)

Time to find a fault when the main process is in a given state can be defined as a recursive relation (23). The equation incorporates value of time to find a fault for preceding state.

\[ Y_k = X_k + q(k-1|k)Y_{k-1} \]  

(23)

where:
\( Y_k \) – random variable which expresses time to find a fault while the process is in state \( k \);
\( X_k \) – random variable which expresses time of being in state \( k \);
Probability distribution for the $Y_k$ variable can be determined by computing a convolution of probability densities of independent random variables which are components of the sum in equation (23). Based on Borel’s convolution theorem, equation (24) is determined by applying Laplace transform and further simplifying transformations.

$$G^*(s,k) = \prod_{j=0}^k F^*(s \cdot \prod_{i=j+1}^k q(i - 1|i,j)).$$

(24)

where:

- $G^*(\cdot) –$ Laplace transform of probability density function $g(\cdot)$ which expresses time to find a fault while the main process is in a given state;
- $F^*(\cdot) –$ Laplace transform of probability density function $f(\cdot)$ which expresses time of being in a given state.

An absolute time to find a fault, irrespective of a state in which the main process currently resides, is determined based on equation (25).

$$Y = \sum_{k=0}^n p(k) \cdot Y_k$$

(25)

By applying Laplace transform for equation (25) and using equation (24) finally equation (26) is received.

$$K^*(s) = \prod_{k=0}^n \prod_{j=0}^k F^*(s \cdot p(k) \cdot \prod_{i=j+1}^k q(i - 1|i,j))$$

(26)

The moment generation feature of Laplace transform or an inverse transform calculation can be used to get formulas that are used to determine expected value (27) and variance (28) of time to find next fault in the main process.

$$E(Y) = \sum_{k=0}^n p(k) \cdot \sum_{j=0}^k \prod_{i=j+1}^k q(i - 1|i) \cdot E(X_j)$$

(27)

$$V(Y) = \sum_{k=0}^n p(k)^2 \cdot \sum_{j=0}^k \prod_{i=j+1}^k q(i - 1|i)^2 \cdot V(X_j)$$

(28)

8. Model verification

Presented in previous chapters approach to software reliability verification during validation process was practically applied on data collected during realization of a real software development process which aimed to enhance functional capabilities of a complex, real-time system. Four independent teams were working on new version of the system, implementing separate functionalities. Therefore, from the proposed model perspective there were four individual production processes for which reliability analysis with use of maturity and readiness to integration indicator as well as cost function was performed. Also the method of determining reliability characteristics for a software under parallel testing performed by multiple teams was verified.

To get evaluations for Musa-Okumoto model parameters, current intensity of faults and expected number of software faults in a given time perspective, the SMERFS3 application was used as a tool which has proved its usefulness in software reliability researches [12].

Analysis of importance of system modules, necessary to determine weights of the modules and then the reliability evaluation risk factor value, shown that the applied complexity metrics in combination with functional criticality of a module are very effective in terms of ability to project the real situation in the evaluated system. Among forty seven modules, ten of them got weight 4, nine of them got weight 1 and the rest of modules got weight 2. Such a result also reflects reasonable architecture of the evaluated system in terms of program structure.

The length of a time step used during analysis was 24 hours. For each time step values of used factors were updated based on information concerning progress and results of the validation process. Picture 3 presents waveform of a function reflecting changes of values of the reliability evaluation risk factor over all 65 time steps for one of the individual processes, with two different values of the constant $\alpha$ from equation (8). Value 1,15 reflects quite low believe of a decisive person about the significance of deviation between number of faults expected and number of faults discovered, while value 1,5 reflects rather serious believe about such a fact. As can be seen from the picture, in the latter case the counted values of $H(k)$ automatically expresses higher level of uncertainty about the evaluated software reliability and the wave is less linear due to
longer period when the risk factor value reduces slowly.

![Graph showing reliability evaluation risk factor value changes](image1)

**Fig. 3. Reliability evaluation risk factor value changes**

The Netica application, being a tool which has proved its usefulness in researches requiring probability inference [13], was used to create the Bayes net presented in figure 1, in order to evaluate number of faults introduced into the software under testing. To build appropriate formulas to be used to determine values of parameters of probability distributions for the nodes “Produced Code Size” and “Number of Introduced Faults”, relevant information from historical projects were used. Having the expected number of introduced software faults, the fault density value was updated after each time step and on this basis value of the fault density adequacy factor (15) was determined. Changes of value of the factor over all 65 time steps for one of the individual processes, called “process 4”, is presented in figure 4.

![Graph showing fault density adequacy factor value changes](image2)

**Fig. 4. Fault density adequacy factor value changes**

Analysis aiming to compare results of applying into the decision making process the reliability evaluations produced by the selected SRGM and the indications given by the maturity and readiness to integration indicator was performed weekly. At the same time already incurred costs as well as forecasted costs were tracked, based on the cost function changes (16). Use of the maturity and readiness to integration indicator proved purposefulness of application of such a method under the circumstances when level of achieved software reliability is treated as a crucial criterion used by a decisive person to make key decisions in software development project. For one of the analyzed individual processes prolongation of the validation process until the maturity and readiness to integration factor reached satisfactory level, enabled to reveal by 43% higher number of faults in the software and to reduce by that the overall cost in the projected time horizon by almost 60%. It was assumed that the faults that could have been detected during the prolonged period of the validation process can be treated as a predicted number of faults from the formula (16) perspective, when additional time \( x \) is sufficiently long e.g. a year.

The method of reliability characteristics determination for the main process, presented in chapter 7, proved usefulness of the described approach.

Figure 5 presents changes, over all 65 time steps, of values of the probabilities of being in a particular state of the main process. The probabilities were determined based on formula (22). As there were four individual production processes, there were 5 states defined for the main process, according to principle from chapter 7. Necessary value of \( \mu \) was determined by experts working for a project and it was constant along the validation process (\( \mu = 0.00641 \)). Necessary value of \( \lambda \) was determined based on faults detection data and it was a subject to update before each of the planned reliability analysis sessions (weekly), see table 1.

**Tab. 1. Values of parameter \( \lambda \) used for the consecutive reliability analysis weekly sessions**

<table>
<thead>
<tr>
<th>Time step (k)</th>
<th>( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0.000989</td>
</tr>
<tr>
<td>14</td>
<td>0.000585</td>
</tr>
<tr>
<td>21</td>
<td>0.000378</td>
</tr>
<tr>
<td>28</td>
<td>0.000316</td>
</tr>
<tr>
<td>35</td>
<td>0.000229</td>
</tr>
<tr>
<td>42</td>
<td>0.00022</td>
</tr>
<tr>
<td>49</td>
<td>0.00021</td>
</tr>
<tr>
<td>56</td>
<td>0.000171</td>
</tr>
<tr>
<td>63</td>
<td>0.000131</td>
</tr>
<tr>
<td>65</td>
<td>0.000129</td>
</tr>
</tbody>
</table>

As it can be seen from figure 5, as time passes it is most probable that none of the individual
processes is in fault fixing state, what is a logical consequence of elimination of consecutive faults from the software.

![Graph](image)

Fig. 5. Unconditional probability of being in particular state of the main process value changes

Figure 6 presents changes, over all 65 time steps, of value of the time to fault detection expected value $E(Y)$, determined according to formula (27), including also value of the standard deviation $D(Y)$, counted as a square root of the variance $V(Y)$ determined according to formula (28). Values of $E(X)$ and $V(X)$ used in formulas (27) and (28) respectively, were counted based on the fact that random variable was assumed to be exponentially distributed. In such case, the expected value is represented by inverse of the rate parameter and the variance is represented by inverse of the rate parameter to the second power. Value of the rate parameter was a known value. As it was expected time to find next fault in the main process was successively longer and longer, reflecting the fact of continuous reduction of number of faults in the software.

By comparing received results of the time to fault detection expected values for the examined process $E(Y)$, it turned out that applied method generates results very similar to approximation performed by model Musa-Okumoto, what is depicted in figure 7. Approximation by Musa-Okumoto is described as "M-O (aprox)". The real observed times between consecutive faults detection in the main process is described as “TBF main process”.

![Graph](image)

Fig. 6. Time to fault detection expected value in the main process value changes

![Graph](image)

Fig. 7. Time between consecutive faults detection in the main process

9. Summary

Proposed model showed reasonable results of its practical applicability for the analyzed data set consisting of test related as well as system related information. Performed analysis shows also perspectives for potential improvements of the proposed approach to software reliability examination for complex systems. Additional attention shall be put on the results of application of the method described in chapter 7. As well, an attempt to develop more complex method for determining the maturity and readiness to integration indicator value is to be taken. That are the preferred directions for further researches on the field of software reliability examination.
10. Bibliography


Wieloaspektowy model predykcji niezawodności oprogramowania w procesie testowania

R. PELKA

Badanie niezawodności oprogramowania stanowi istotną część realizacji planu jakościowego w procesie produkcji oprogramowania. Poprzez monitorowanie zmian wartości prognozowanej niezawodności oprogramowania w odniesieniu do założonych celów jakościowych można dokonywać analizy bieżącej sytuacji oraz w razie konieczności podejmować kroki sprzyjające realizacji założonego planu. Wykorzystanie w celu predykcji niezawodności jedynie modeli wzrostu niezawodności oprogramowania, bazujących na historii wykrywania błędów w badanym oprogramowaniu, wydaje się być podejściem zbyt uproszczonym. Podejście to w pewnych okolicznościach realizacji procesu walidacji oprogramowania może być obarczone dużym błędem i wpływać na podejmowanie błędnych decyzji przez decydentę. W związku z tym, w zaproponowanym modelu wykorzystuje się szereg dodatkowych informacji o testowanym oprogramowaniu oraz samym procesie walidacji w celu uzyskania bardziej wiarygodnych efektów analizy niezawodnościowej, będących jednocześnie odpowiednią informacją zwrotną dla decydenty z punktu widzenia założonych realiów prowadzenia projektu programistycznego. Integralną część prezentowanego podejścia stanowi aspekt wyznaczania charakterystyk niezawodnościowych systemu testowanego równolegle przez kilka niezależnych zespołów.

Słowa kluczowe: niezawodność, oprogramowanie, modelowanie, testowanie.
Security of Statistical Databases as an element of Enterprise Security Architecture

Ł. ŚLEZAK, J. BUTANOWICZ
lslezak@wat.edu.pl

Faculty of Cybernetics, Military University of Technology
Kaliskiego Str. 2, 00-908 Warsaw, Poland

The protection of statistical databases is the problem with the highest importance for many organizations, in particular the public administration. An appropriate method for statistical databases protection has to ensure a balance between safety and quality of data, taking into account the associated risks and business needs. To improve security of statistical databases integration of protection mechanisms with the enterprise security architecture (i.e. using SABSA methodology) is proposed. Coherent and comprehensive approach of the protection is a result of the proposal. It allows statistical databases safety to be in line with business requirements and ensure that statistical databases security mechanisms are integrated with existing enterprise security mechanisms and services.

Keywords: enterprise security architecture, SABSA.

1. Introduction

Statistical databases are used to store anonymous data. This data is available to the user only at the level of statistical groups, and reserved at a unit-level. The main difference between the statistical databases and relational databases is a very limited set of allowed queries. Inquiries are mainly limited to operations such as the number of a set, the sum, average, and several other statistical calculations that are performed on a subset relationship.

The problem of the statistical databases protection is the provision of statistics on people, while limiting the ability to access information about any specific individual. Achieving this is extremely difficult, because each statistic contains derivative information on the basis of which was calculated and, therefore, any provision of accurate statistics is a risk factor of disclosing private information [2]. In addition, the correlation of different statistics may lead to the disclosure of exact or approximate information of private data. An additional difficulty is the protection of sensitive personal data from the user having knowledge of the characteristic values of an individual acquired outside the statistical database [6].

Advanced researches in the field of protection of statistical databases, as well as existing solutions usually involve two basic approaches that are often used in combination. The first approach is to restrict access to database systems where access is granted only to authorized and authenticated individuals to carry out the approved analysis. Additional restrictions may also be introduced at the level of the results of statistical analysis, making it impossible to publish outside of a statistics organization (i.e. Eurostat, the statistical office in France, the Netherlands, Denmark, or Sweden) [13]. The second approach is to reduce or replace data, where less than a full set of data is published, or the data has been distorted in some way before publication. There are many methods of statistical databases by manipulating the data that are the object of analysis (while maintaining statistical accuracy of resulting analysis). The importance of the protection of privacy in statistical databases is growing nowadays in Poland mainly because of the national digitalization of health care and IT support for the Agricultural Census 2010 and National Census and Dwellings 2011.

For a coherent and comprehensive approach to the security of statistical databases, an integration of the existing security mechanisms with the enterprise security architecture is advisable. The security architecture is a coherent set of principles, guidelines, benchmarks and standards of security, along with a description of their interconnection, and taking into account the circumstances and the needs of business organizations. To implement the security architecture is to provide adequate mechanisms for business needs, which will reduce the risk of data breaches.
An example of the methodology used to manage the security architecture is SABSA. (Sherwood Applied Business Security Architecture) SABSA methodology is used by many organizations all over the world to develop the security architecture, in particular in the public administration. An example would be the United Kingdom’s Ministry of Defence, which has chosen the SABSA methodology to implement Defence Information Security Architecture [15].

SABSA methodology presents a business approach to security within an organization. It is used for a comprehensive management of security service and security architecture. It allows the use of well-known standards as components of the architecture in a consistent manner, which together provides a comprehensive model, based on business requirements.

By using the SABSA methodology it is possible to develop security mechanisms that can guarantee the security of statistical databases in a specific business organization. At the same time, such an approach ensures the integration of security mechanisms for statistical databases with other security services and mechanisms used in the organization.

In this paper, we propose an integration of statistical database security mechanisms with enterprise security architecture via SABSA methodology. In accordance with the principles of creating the security architecture the paper covers all SABSA layers. In following chapters a model overview is presented starting from the top layer – business requirements and going down to lower layers – the implementation of the security mechanisms of statistical databases, and elements of the operational layer.

2. SABSA Model

The SABSA model is based on six layers:

- Contextual – a description of the business and business requirements
- Conceptual – a high-level description of security
- Physical – security services
- Logical – mechanisms and security methods
- Component – implementation of the selected security method
- Operational – monitoring changes in the security architecture and security audits.

Operational layer is defined in parallel with the logical, physical and component layers. In each of these layers six aspects of security are defined, which form the SABSA matrix. The SABSA matrix structure is analogous to the Zachman framework [14], but it should be noted that they were developed independently. Creating enterprise security architecture is to develop various aspects of each layer starting from the highest to the lower ones.

The approach presented in this document is based on a layered SABSA model. Selected SABSA matrix elements are used, because not all elements need to be addressed. Selected elements fully show the ease of use and usefulness of the SABSA methodology in order to integrate the security of statistical databases with the security architecture.
Analysis presented in the paper are carried through all SABSA layers from top to bottom. The results of the analysis of items resulting from the previous layer are a direct source of the development of elements of subsequent layers. It should be emphasized that the result of the elaboration of the various elements of each layer will vary for different organizations.

**Contextual Layer**

The contextual layer represents the business view on the organization’s security. The type of security architecture that organization needs is defined in this layer. The aim of the contextual layer is to collect business requirements for security and create a business risk model for statistical databases.

The first step to create the security architecture is to define business requirements. Business requirements are all important aspects of security from the perspective of the organization. Defining the business requirements must be carried out, particularly in the following areas: Legal requirements – the most important source of business requirements are legal provisions in force within the country. The Polish constitution of 2nd April, 1997, introduced into the Polish legislation the principle of the protection of personal data. The principle was specified by the act of August 29th, 1997 on the protection of personal data, which is an implementation of the directive of the European Parliament and the European Council of October 24th, 1995 on the protection of individuals with regard to the processing of personal data and the free exchange of such data. Persons whose data is being processed were provided with instruments to control the process of processing of their data, while the database administrators imposed a series of obligations that maintain data security.

In the face of progressive informatisation of society, the development of new technologies and globalization issues, data safety is one of the major challenges of modern information science. It should be noted that in addition to personal data, there are other types of data, for which protection is especially important for those affected. An example could be any confidential company information. It is to protect the statistical database that provides any kind of data. However, in view of the critical safety of personal data, they are most important.

In addition, every organization is forced to comply with the laws specific to their business. An example would be the Polish Law on the National Census, where in cooperation with the Polish General Inspector of Data Protection introduced a lot of requirements to increase data security:

- The Polish Central Statistics Office may entrust personal data to other entities
- Introduced additional security of information systems used during the census
- Prohibited from sharing personal data by the Central Statistical Office for purposes other than statistical.

It is worth stressing the latter points. The Polish Central Statistical Office is authorized to release personal data exclusively for statistical requirements in the absence of adequate (in the statistical sense) secure database entities to which data are available. This is an example of the lack of an architectural approach to data security i.e. spot data protection, without taking into account the comprehensive approach to ensure real security.

- Business Strategy, including the mission, vision and objectives of the organization, which can be found, among others in:
  - State Informatisation Plan for years 2007–2010
  - Polish National Security Strategy
  - National Development Plan.
- The key resources that are relevant to the organization and must be protected by appropriate safeguards for business information and business information systems.
- Business processes to which it is necessary to implement information security. These are all processes related to:
  - Business interaction
  - Business transactions
  - Business communication.

A complete set of business requirements is a contribution to the creation of a business risk model. Creating the business risk model is to verify the business requirements for meeting them through a standard database. In each case, if the requirement is not met the risk corresponding to this requirement has to be defined, and then put to the business risk model.

A full risk model describes all the business risks associated with the use of standard databases to provide statistical data. One can easily draw the conclusion that all risk mitigation is a key to the implementation of security mechanisms to protect statistical data. Risk mitigation drives an implementation of the security mechanisms protecting statistical data.

The impact and the vulnerability are described for each threat. The final risk
evaluation is the product of both components. This value determines the priority with which the risk should be mitigated. Each organization must be examined in order to select the value below, which the risk may be regarded as acceptable.

**Conceptual Layer**

Within the conceptual layer the business requirements are translated into a security concept. At this stage, the concept of a security method that meets business needs is created. The concept represents the vision of full security, and not its individual components.

Methods of statistical database protection affect the value of the obtained statistics. In addition to the security relationship between the data and the accuracy of the statistics obtained, an important element is the cost of implementing the method and its impact on the effectiveness of the database.

In order to define all the relevant factors affecting the quality of the operation of the statistical database, the element of the conceptual layer is used, i.e. the business attributes profile. The business attributes profile is a set of attributes describing the concept of security – in our case, factors affecting the quality of the operation of statistical databases. Profile results directly from business requirements. Normally, global standards like ISO 17799/27001 defines key business attributes: integrity, availability and confidentiality. SABSA methodology defines 85 business attributes for the entire organization. In accordance with best practices and global studies, there shall be five main attributes for statistical databases: security, information loss, cost, bias, consistency. Please note that it is possible to describe regular databases with a larger set of attributes, but only specific attributes for statistical databases were selected. Other attributes should be taken into account when creating the security architecture (for other elements of the organization).

Attributes specified for the purpose of statistical database security are selected and described below. First three attributes are of the highest importance from the business requirements perspective. Last two attributes are specific to methods based on data perturbation. Each proposed attribute is presented along with a measurement method. According to business requirements the database administrator, who is responsible for information assurance, sets target performances of every business attributes.

Measurement methods of these attributes are used for verification if the target performance criterion is met. Attributes:

- **Security** – a business attribute describing a statistical database system ability to mitigate the risk of compromising confidentiality. The measuring method selected for the Security business attribute is a combination of the following methods: Distance-based Linkage Disclosure (DLD), Probabilistic Linkage Disclosure (PLD) and Interval Disclosure (ID) as defined in [10, 11]. The combination of these methods is constructed as follows: Security = 0.25 (DLD) + 0.25 (PLD) + 0.5 (ID). The measurement method described for the Disclosure Risk Assessment in [10] is proposed for Security business attributes among other defined and proposed metrics in literature because of its general approach and because a lot of security methods have been evaluated using that combination.

- **Information Loss** – a business attribute describing the difference between original data and data released to the legitimate user. Information Loss stands for quality and usefulness of released data. Measuring the method proposed for the SABSA methodology is an application of Information Loss Measures from [10, 11]. The method is based on averaging the mean variations of differences between original and released matrices representing original and released datasets respectively, as well as of differences of their covariance matrices, diagonals of covariance matrices, vectors of variable averages. In the method the mean absolute error of the difference between correlation matrices of original and released matrices is taken into account.

- **Cost** – a business attribute that reflects cost of an initial implementation and cost of overhead processing per query. The measuring method measures an extra effort consumed on the initial implementation of the protection mechanism expressed in money and a difference in time consumed on processing a given set of queries on the database with the protection mechanism and without. Importance of any of these two factors is organizational specific.

- **Bias** – a business attribute describing how much actual value of a statistic differs from a perturbed value returned by the statistical database. The measurement method
proposed is to empirically create a set of the most popular statistic queries (based on historical logs) and use that set for checking the variance between original and perturbed statistics.

- **Consistency** – a business attribute that represents the lack of:
  - **Contradictions** – i.e. repetitions of the same query yielding different results, or when an additive statistic corresponding to a row or column sum differs from that obtained by adding the statistics in the row or column of the table.
  - **Paradoxes** – negative values are returned for counts. Proposed measuring method is based on knowledge about a specific protection mechanism. If the specific mechanism makes possible creation of contradictions or paradoxes set value 1 for the method, otherwise 0.

Another key element of a conceptual layer is to develop the control objectives, from which one can reduce the risk (from the business risk model in the contextual layer) to an acceptable level (defined also in the contextual layer). For each risk from the model, control objectives are defined. Typically, the control objectives are selected from the standards such as COBIT, ISO 17799, NIST 800-53. In that case, the objectives are so unique that we need to define it according to a business risk model. From the list of control objectives the set of control objectives that must be met to secure statistical databases in accordance with business requirements will be created. In addition, it is the basis for a security audit (as part of the operational layer).

**Logical Layer**

Logical layer contains a comprehensive description of the organization’s functional requirements to security resources. The logic of security is defined independently on the mechanisms that implement it as defined in the physical layer.

The main product of this layer is to define the security services – a set of activities for security. Compliance with security services will ensure security in the organization at the required level (arising from the business requirements). In our case, the service is to protect statistical databases, taking into account the compliance with control objectives and implementation of business attributes.

This layer describes in details the problem of the protection of statistical databases – how to create a slice of the security architecture for a statistical database. In addition, the link between the upper layer products with the proper implementation of the security method is created.

**Physical Layer**

The physical layer covers the selection of a specific security method to be used in order to meet the requirements of the organization. The main products of the physical layer are security mechanisms – different ways of realization of security services defined in the logical layer.

There are many mechanisms for the privacy protection of statistical databases (while maintaining statistical accuracy of derived statistics), which can be divided into four main groups:

- Query set reduction
- Data perturbation
- Result based perturbation
- Others.

In order to design a security method that addresses the security service as defined in the logical layer, analysis of hybrids of security mechanisms that ensure that the control objectives are met, must be performed. The next step is to evaluate hybrids in accordance to the business attributes profile.

The challenge is to ensure a high level of data security, while maintaining satisfactory accuracy of the statistics calculated. For this purpose it is necessary to analyze various mechanisms and a combination of selected mechanisms into the security methods for achieving this goal.
Fig. 2. A classification sample of basic mechanisms

Fig. 3. Example of a model to compare the different hybrids of security mechanisms. It is assumed in the model that weights of all business attributes are equal.

The weight of each business attribute has to be set depending on organization business requirements. Security and Information Loss are attributes that are opposite to each other so their importance are supposed to be equally weighted. Cost attribute is the third most important business attribute and its weight fully depends on organization resources.

It should be noted that the expected value of each attribute is defined in the business attributes profile in the conceptual layer. The final product of the physical layer is the selection of a specific method – hybrid of security mechanisms that ideally suits the needs of business organizations.

Component Layer

The component layer covers the selection of the implementation technique of the security method, which will be used to secure data. Implementation of the security method that has been selected in the physical layer takes place in the component layer. Implementation can take place in various ways. The selected implementation technique of the security method must work together to ensure consistency and completeness of the security infrastructure.

The product of this layer is consistent with the requirements of business organizations and will be used for statistical data provision.

Operational Layer

Operational layer describes the processes and procedures through which information systems of organizations operate in a safe manner. The operational layer is defined
in parallel with the logical, physical and component layers.

The first element of the operational layer is to define a process of monitoring changes to the business requirements and business risk model. On the basis of these changes, the security architecture is updated, and thus, also the implementation technique of the security method developed for statistical databases must be updated.

The second important operational element is to carry out periodic safety audits of statistical databases, in accordance with the control objectives defined in the conceptual layer. In case of discrepancies specific actions should be introduced that would allow to meet the control objectives.

3. Summary

The aim of this paper was to propose integration of existing mechanisms securing statistical databases with the security architecture of the organization based on SABSA methodology.

A coherent and comprehensive approach is ensured this way. It allows the integration of the security method developed for statistical databases with existing security mechanisms and services of other architectural elements of the organization. In addition, it allows for a security method development that complies with business requirements – sometimes ensuring the maximum value of each attribute is not consistent with the requirements of an organization.

Defining the lower SABSA layers allows increasing details of the safety aspect. On the other hand, implemented security mechanisms can be linked to business justifications by moving towards top layers.

Future work will be focused on finding appropriate methods for securing statistical databases. Verification and analysis of proposed techniques is not only based on properties of these methods, but also their applications to specific databases. The usability of methods highly depends on database structure and characteristics of the data it contains. The challenge is to find methods of ensuring a balance between safety and quality of data for a statistical database, taking into account database applications and the associated risks.

4. Bibliography


Bezpieczeństwo statystycznych baz danych jako element architektury bezpieczeństwa

Ł. ŚLĘZAK, J. BUTANOWICZ

Ochrona statystycznych baz danych jest problem najwyższej wagi w wielu organizacjach, w szczególności jednostkach administracji publicznej. Odpowiednia metoda ochrony statystycznych baz danych musi zapewnić równowagę pomiędzy bezpieczeństwem oraz jakością danych, biorąc pod uwagę ryzyko i wymagania biznesowe. Najlepszym sposobem osiągnięcia tego celu jest integracja mechanizmów ochrony statystycznych baz danych z architekturą bezpieczeństwa przedsiębiorstwa (np. z wykorzystaniem metodologii SABSA). W ten sposób zapewnione jest spójne i kompleksowe podejście. Pozwala to na wdrożenie mechanizmów ochrony statystycznych baz danych zgodnych z wymaganiami biznesowymi i zapewnienie, że te mechanizmy są zintegrowane z istniejącymi usługami oraz rozwiązaniami z zakresu bezpieczeństwa.

Słowa kluczowe: architektura bezpieczeństwa, SABSA.
Clustering method based on the analysis of the access request stream in object-oriented databases

K. TOMASZEWSKI
ktomaszewski@wat.edu.pl

Institute of Computer and Information Systems
Faculty of Cybernetics, Military University of Technology
Kaliskiego Str. 2, 00-908 Warsaw, Poland

Recent studies on modern database management systems consider object-oriented databases as a possible significant extension of the modern database functionalities. However, new functionalities bring an increased processing complexity. This may also result in an increased demand for computing resources and the deterioration of database performance. In this article, the use of clustering methods has been described as one of performance techniques in object-oriented databases. The article includes an introduction to the popular clustering methods discussed so far. Afterwards, a new method has been introduced in order to analyse the access request stream as the basis for a new clustering approach in OODB. Graph techniques are discussed as the basic elements of the familiar clustering methods and their adaptation to the aforementioned new approach. This article also describes limitations of the existing methods and the possible impact on the new technique. Finally, selected algorithm variants are proposed for the new clustering method to improve performance of object-oriented databases.

Keywords: clustering, object-oriented database, OODB.

1. Introduction

Research and development of databases is motivated by the need for better support of design and maintenance of modern information systems. Object-oriented databases are one of the significant threads of this research as they bring a number of new features to modern database solutions. Unified meaning of an object-oriented database system is still open to discussion. However, the general principle is to combine the features of object-oriented programming languages, such as data model abstractions, with database features, such as the possibility to persist the data. The object-oriented data model extends the capabilities of the relational data model, i.e. with the use of complex data attributes. Simple column values, known from relational databases, can be extended by simple, as well as complex object attributes, such as sets, lists or references to another object instances. One of the most significant innovations is the introduction of object identity as a way of accessing the specifically desired data instance with the use of reference attributes.

2. Performance of object-oriented databases

Extended capabilities of object-oriented databases increase, however, the general complexity of database management systems. This may also result in an increased demand for computing resources required for processing database operations. Deterioration of general database performance may involve an increased time of serving the requests incoming from database users. This factor motivates the design and development of a wide variety of performance techniques for object-oriented database management systems.

In the bibliography, we can find numerous descriptions of research aimed at improving the effectiveness of the aforementioned class of systems.

Selected aspects of general performance research were described in [1]. They include clustering, partitioning, replication of data, as well as concurrent processing or indexing techniques in data access processes.
3. Data Clustering

Data clustering techniques have been selected from the above-mentioned performance research as a subject for further analysis. Clustering itself includes a wide scope of knowledge that preceded the creation of first computers. Modern definitions of clustering mainly relate to the issues of cluster analysis and identification of thematically interrelated groups of entities. Cluster analysis is based, however, on statistical classification algorithms which assign statistical observations to the identified classes, based on the characteristics of such observations.

The specific nature of database management systems provides an opportunity to use the information from the cluster analysis, based on fact that single objects within a database are usually grouped together and stored in larger, separate logical units. This phenomenon has been implemented in two popular database organization models.

![Buffer memory model](image1.png)

**Fig. 1. Buffer memory model**

The first one is a model of a centralized database management system. It is based on a single database unit, serving all incoming access requests. In this model it is assumed to serve access requests with the use of buffer memory (cache, RAM), as well as storage memory (HDD) of the database system. Objects stored in the storage memory are grouped in memory pages as a larger logical memory units (figure 1). Memory pages are used as the basic data exchange units between the buffer and the storage memory. Serving of incoming access requests involves loading the whole page, which contains the referenced object, into the buffer. If the subsequent access requests refer to the objects, stored within the same page, they will be served with the use of buffer only (no further HDD read will be necessary).

![Related objects stored together in logical memory units](image2.png)

**Fig. 2. Related objects stored together in logical memory units**

The aim of clustering, in general, is based on the assumption that the subsequent incoming access requests will refer to objects with identified similarities, in terms of a particular similarity criterion. The goal is to provide an effective similarity criterion, embedded in a complete clustering approach, in order to identify groups and store them in correct logical units (figure 2).

The second model, which involves grouping data into bigger logical units is a distributed database model. In such models, data objects are distributed among a number of database nodes, operating in a shared network structure. An access request, incoming to a particular database node, will be served with the use of data stored locally and if necessary, additional requests to the remote nodes will follow.

Effective strategy for object location among the database nodes will result in serving the subsequent access request with the use of local resources, while reducing the number of remote accesses. The second model is a higher-level abstraction, which includes also a previous buffered-memory model for every single database node.

4. Existing clustering techniques

Recent work related to similarity evaluation has brought a comprehensive variety of techniques for analysing entity properties.

One of the general approaches is to define a measure and evaluate entity similarity based on selected entity attributes [2], [6].

Examples of more specific object-oriented methods assess the object similarity from the data model description, e.g. inheritance dependencies [3], [4], [5] and by analysing associations between the actual object instances [6], [7]. It is also possible to examine a set of predicates fulfilled by particular data instances [9]. These methods can operate in the data model...
domain, as well as data instance domain. Other examples of the method are able to examine the popularity of data entities in order to perform effective clustering (so called HOT-COLD techniques) [8].

Most of the existing clustering techniques can be described with the use of a general clustering algorithm presented below.

First, essential entity features (attributes) are selected, catering significant information for further similarity deduction. Examples of such features include values of chosen attributes, as well as a mutual object-oriented class inheritance hierarchy or common sets of fulfilled query predicates.

Next, the representation format of the selected features is adapted to serve as an input for a chosen clustering tool. At the same time, a similarity measure needs to be defined. This measure is usually specific to a particular clustering tool. With a defined similarity measure, the chosen clustering tool may be used to identify clusters in the object set. Examples of popular clustering tools include such methods as the k-means method [10] or hierarchical clustering [11]. Outcome results can be then analysed and information about identified clusters may be used for the purpose of reorganisation of the database.

5. Access request stream analysis

In addition to the existing ones, a new clustering technique has been introduced. Unlike the previously described techniques, the new technique does not involve an analysis of internal object features. It is based, however, on the observation and analysis of the access request stream incoming to the database system. The goal of the analysis is to identify groups of objects which are accessed in relatively short time intervals. Proximity of access requests in time is then used as a grouping criterion. In the implementation of the method, recorded information regarding the last access time for each object instance is used. Based on such information, various groups of related objects are identified (figure 3).

![Fig. 3. Identification of object group in access request stream](image)

Identified object groups are then placed on the memory pages (figure 4).

Observation of the access request stream is the only source for the object grouping deduction. Low-level input data simplifies the implementation method, as compared to other popular clustering methods which are based on complex analysis of data models or analysis of associations between object instances.

![Fig. 4. Data reorganization](image)

If the objects referenced by the subsequent access request are placed on the same memory page, there is no need to load a new page into the buffer. As a result, the overall page swapping process will be reduced in the database management system processing.

6. The use of CFNG graph

For the purposes of the approach described above, a clustering tool needs to be provided in order to perform actual clustering into proper groups. From among a variety of clustering tools operating in metric space, the Colored Farthest
Neighbour Graph method (CFNG) has been selected, as it introduces some additional features described in chapter 9 of this paper. The CFNG graph reflects the relation of the farthest neighbourhood according to a chosen distance measure. The Colored Farthest Neighbour Graph method has been originally described by A. Hausner in [12]. It originates from the Shared Farthest Neighbour Graph method, described by S. Rovetta and F. Masulli [13].

At first, a similarity measure should be defined or more precisely, a distance measure between objects being clustered. As the approach involves grouping objects with the close requests occurrences, the intuitive choice is to determine the distance measure $d(O_m, O_n)$ as a difference of time values between access requests and the objects $O_m$ and $O_n$ in the linear time domain (figure 5).

For each object in the analysed set, the distance to any other objects is evaluated. Based on the measures, the Farthest Neighbour Graph (FNG) is created then. In the graph, the vertices represent particular objects. Each edge connects the object to the one with the maximum distance value. This means that at least one of the objects connected by the edge is the farthest neighbour of the other, according to the defined distance measure (figure 6A). Afterwards, the resulting graph is subjected to the colouring process. In [12] A. Hausner shows that the FNG graph does not contain any cycles, thus it is always a tree structure (figure 6B). As a result, a two-colour model is always present in the graph (figure 6C).

By colouring, each node is given a label which assigns it to one of the sub-clusters. As a result, we arrive at a two-colour graph (figure 6D). Each iteration of the algorithm divides the processed set binary into two subsets.

The stop-condition of the algorithm can be determined in various ways. The number of iterations may depend on the expected properties of the final clusters or just on the expected number of the clusters.

It is also possible to construct a full cluster hierarchy, by repeating the binary division until the result of N-number of single-element subsets of the N-element initial set.

Fig. 6. The CFNG Method: A. Setting the edges; B. Tree structure; C. Graph colouring; D. Identified clusters

Usually, size features of the objects and the capacity of the cluster containers are determined. In this case, the clustering method can be executed until every outcome cluster can be stored in a single cluster container.

7. CFNG method limitations

Several situations can be described where the CFNG method can result in poor quality clustering. Although it is certain that the steps of the algorithm can be properly executed in each case, it cannot be ruled out that the algorithm itself will reflect the natural properties of the observed clusters in an incorrect manner. The literature on the subject [12] describes a situation susceptible to incorrect algorithm outcome. It results from the binary-division feature of the method. In each algorithm iteration, every element of the examined set is assigned to one of two sub-sets. These sub-sets reflect the natural “poles” of the set (based on the most remote points). If the analysed set contains a group of elements located near the middle point between the poles, the algorithm will break down the middle group by assigning each element of it to one of the two border clusters (figure 7).
Such division is called *hasty split* and is also possible for other clustering methods. The results of low quality hierarchical divisions have been described in detail in the literature on the subject. A number of repair, preventive and mixed techniques have been also proposed to counteract low quality divisions.

8. **Simplification of the algorithm**

Significant simplification can be observed when CFNG method operates in the linear metric space. In each iterations it is possible to distinguish two border objects: $A$ and $B$ with the lowest and the highest position value in linear space. Additionally, the mid-point $m$ can be also distinguished in the space between the extreme points dividing it into two equal areas (figure 8). It is possible then to assign every object to one of the sub-clusters using simple inequality. All the objects with the position value lower than the mid-point will be assigned to one cluster with the extreme maximum object as the farthest neighbour. By analogy, all the objects with the position value higher than the mid-point will be assigned to the second cluster with the extreme minimum object as the farthest neighbour.

Simplification however does not relieve the method of the previously described “hasty split” among the clusters located near the centre of the set in the space of time.

9. **Algorithm modifications**

Modular structure of the clustering algorithm provides an opportunity to replace the distance measure between the objects, while using a CFNG graph as a clustering tool. A significant feature of the CFNG is that it enables the clustering of objects without knowledge of their location in metric space. The distance measure may be then defined as an abstract value resulting from the other object model features. An example of an abstract distance measure, which may be used as a substitute, is the aforementioned feature of affiliation of object classes to common inheritance hierarchies [3], [4], [5]. Furthermore, other distance measures, for example vector methods – taking into account a greater number of feature dimensions – can be considered as replacement candidates.

Moreover, the possibility to present the location of the object in the linear time domain creates an opportunity to use such information as an input data for other clustering methods. The CFNG graph construction method can be then easily replaced and compared to other methods during actual evaluation experiments.

10. **Summary**

This article refers to the growth in complexity of processing in object-oriented databases. This growth is related to the introduction of new features and functionalities to modern database systems provided by object-oriented databases. Main innovations are mostly related to the object-oriented data model. The complexity growth affects, in an adverse manner, the efficiency of serving access requests incoming to the object-oriented database. This article describes the search and evaluation of efficiency techniques designed for this class of systems. The following parts elaborate on clustering as one of familiar efficiency techniques used in database management systems. The article contains a concise introduction to popular clustering methods. It also presents a new clustering method, based on the access request stream analysis for object-oriented databases. In addition, the study discusses special features of coloured graphs as a basic tool of the proposed clustering method. Along with the detailed presentation on those features, the paper proposes simplifications and various implementation models of the clustering algorithm. The use of the proposed method in object-oriented databases is aimed at improving efficiency features of those systems.
Metoda klasteryzacji uwzględniająca charakterystyki strumieni żądań dostępu do danych w obiektowych bazach danych

K. TOMASZEWSKI


Słowa kluczowe: klasteryzacja, obiektowe bazy danych, OODB.
INFORMACJE DLA AUTORÓW

1. „Biuletyn Instytutu Systemów Informatycznych” (ISSN 1508-4183) jest czasopismem naukowym, publikującym artykuły z zakresu informatyki oraz badań operacyjnych.
2. „Biuletyn Instytutu Systemów Informatycznych” wydawany jest dwa razy w roku.
3. Przesłane do redakcji artykuły powinny być oryginalne, wcześniej niepublikowane, napisane w języku angielskim.
4. Artykuły powinny być przygotowane zgodnie z INSTRUKCJĄ DLA AUTORÓW.
5. Artykuły są recenzowane zgodnie z PROCEDURĄ RECENZOWANIA.
6. INSTRUKCJA DLA AUTORÓW oraz PROCEDURA RECENZOWANIA są udostępnione na stronie internetowej czasopisma (http://biuletnisi.wcy.wat.edu.pl).
7. Wersją pierwotną (referencejną) czasopisma jest wydanie papierowe.
8. „Biuletyn Instytutu Systemów Informatycznych” jest czasopismem indeksowanym w bazach: BazTech i Index Copernicus Journal Master List.

INSTRUKCJA DLA AUTORÓW

2. Wszystkie nadesłane materiały, tj. pliki w formacie DOC lub DOCX, podlegają wstępnej ocenie merytorycznej przez redaktora naczelnego czasopisma. Redakcja odsyła artykuły, które nie zostały zakwalifikowane.
3. Maksymalna objętość nadesłanego materiału nie może przekroczyć 10 stron formatu A4, przygotowanego zgodnie z szablonem.
5. Jeśli artykuł jest napisany w j. angielskim, to po „Bibliografii” powinien być umieszczony tytuł, streszczenie i słowa kluczowe w języku polskim.
6. Autorzy artykułów proszeni są o zachowanie zasad etyki obowiązujących w nauce. Zgodnie z wytycznymi Ministerstwa Nauki i Szkolnictwa Wyższego, które dotyczą nowego systemu oceny czasopism naukowych, redakcja ma obowiązek informowania autorów o tym, że wszelkie wykryte przejawy nierzetelności naukowej, takie jak ghostwriting i guest authorship, będą demaskowane. Ghostwriting ma zastosowanie w sytuacji, gdy ktoś wniósł istotny wkład w powstanie publikacji, a nie jest ujawniony jego wkład jako jednego z autorów.
Guest autorship to sytuacja, w której autor ma znikomy wkład lub w ogóle wkład nie ma, a pomimo to jest autorem/współautorem publikacji.

7. W przypadku jakichkolwiek wątpliwości, redakcja prosi Autorów o kontakt za pośrednictwem elektronicznej poczty na adres: biuletynisi@wat.edu.pl

**Procedura recenzowania i kwalifikowania artykułów naukowych do wydania w „Biuletynie Instytutu Systemów Informatycznych”**

1. Nadesłany do redakcji autorski materiał naukowy, w formie pliku, musi być przygotowany przez autorów zgodnie z szablonem udostępnionym na stronie internetowej czasopisma.

2. Nadesłany artykuł podlega wstępnej ocenie merytorycznej przez redaktora naczelnego czasopisma. W przypadku gdy artykuł nie jest zgodny z profilem czasopisma lub gdy nie jest przygotowany zgodnie z wymaganiami redakcji, zostaje odesłany do autora.

3. Jeśli artykuł spełnia wstępne wymagania, redaktor naczelny po konsultacji z redaktorem tematycznym przekazuje artykuł do recenzenta specjalizującego się w danej dziedzinie.

4. Lista recenzentów jest podawana do publicznej wiadomości raz w roku, w pierwotnej (papierowej) wersji czasopisma.

5. Recenzenci znają tożsamość autorów, ale autorzy nie znają tożsamości recenzentów. Obowiązuje jednak zasada, aby między autorem a recenzentem nie było konfliktu interesów.

6. Recenzja ma formę pisemną i jest sporządzona na specjalnie przygotowanym w tym celu formularzu.

7. Uwagi recenzenta wymagające dokonania przez autora poprawek są przekazywane autorowi przez redakcję.

8. Zastrzeżenia zgłoszone przez autora do recenzji są przekazywane recenzentom przez redakcję.

9. Po rozpatrzeniu zastrzeżeń autora i odpowiedzi recenzenta na zgłoszone przez autora uwagi, redaktor naczelny podejmuje decyzję o dalszym postępowaniu.

10. Opublikowane mogą być artykuły, które otrzymały pozytywną recenzę lub uzyskały akceptację redaktora naczelnego po rozpatrzeniu uwag autora do recenzji i odpowiedzi recenzenta na zgłoszone przez autora uwagi.