Evaluation of the Software Maintenance Tasks Based on Fuzzy Screening

Zeljko Stojanov*, Vladimir Brtka*, Dalibor Dobrilovic*
* University of Novi Sad/Technical faculty “Mihajlo Pupin”, Zrenjanin, Serbia
zeljko.stojanov@tfzr.rs, vbrtka@tfzr.uns.ac.rs, ddobrilo@tfzr.rs

Abstract— Evaluation of software maintenance tasks can significantly contribute to the efficiency of maintenance management, which is recognized as the most costly part of the software life cycle. This paper presents an approach to evaluation of software maintenance tasks based on fuzzy screening. The construction of data set for the case study is based on the real data extracted from the maintenance repository available in a local very small software company. In the construction of data set participated researchers and experienced software experts from the selected company. Results of the analysis provide evidence that fuzzy screening can be used for evaluation of maintenance tasks based on selected attributes. Further work directions are outlined in the conclusion section.

I. INTRODUCTION

Maintenance has been recognized in literature as the most costly part of software life cycle [1, 2]. Maintenance costs are between 40% and 90% of the total cost of the software life cycle [3]. The aim of software maintenance is to ensure proper functioning of software products after delivery. Software maintenance can occur for many reasons, but maintenance tasks are much more difficult and complicated compared to the tasks during development [4]. The diversity of maintenance processes and activities depends on the domain of software use, software size, frequency of changes, and the limitations in the work schedule, resources and budget [5].

As a result of good practice, Jones [6] identified 23 different tasks that appear in the practice of software maintenance. All these tasks differ in many aspects and have different reasons for being carried out. Because of the complexity of maintenance tasks, it is important to hire skilled engineers and to use state-of-the art methods and tools.

The development of strategies for assessing maintenance activities and maintainability of software products is very important for industrial practice because of high expenses of software maintenance. Maintainability is mainly influenced by two project factors: maintenance task to be performed and people that will perform the task. Actually, small tasks, related to changes in one small module are easier to implement and can be generally classified as simple tasks. On the other hand, larger changes that require modifications of larger parts of software, or even its reconstruction, can be classified as tasks that are more complex. Classification of tasks complexity can be based on structural measures (number of code lines, number of procedures, number of modules) or on subjective assessment of software experts.

Task complexity is mainly examined in literature that is oriented towards information processing, decision-making and job and task design fields [7]. Although task complexity significantly influences the outcomes and costs of the work, the researchers have not agreed about the properties that make a task complex. Campbell [7] treated complexity as: a) primarily a psychological experience, b) an interaction between task and person’s characteristics, and c) a function of objective task characteristics.

If software maintenance is treated as an information-processing task, then the theory of tasks analysis can be applied [8]. Campbell and Carver proposed analytical model that describes the characteristics of maintenance tasks based on Wood’s task complexity model [9]. The model includes internal attributes of software (structural measures), and external attributes of maintenance process (e.g. effort). Anda [10] conducted empirical study in order to evaluate maintainability of four software systems using both structural measures and expert assessment. Anda suggested that there are several examples of potential problems regarding maintainability (effort in maintenance tasks) that were not captured by the structural measures. Although the reliability of expert assessments and judgments are open questions in literature that deals with software maintenance, experienced experts may provide reliable assessments related to maintainability and maintenance task complexity.

The complexity of maintenance tasks is influenced by many factors from human behavior domain (personal characteristics, knowledge, experience) and technical domain (maintainability). On the other hand, the complexity of maintenance tasks influences the internal organization of software organization, the efficiency of services provided to clients and the benefits for both software organization and its clients.

Maintenance tasks are triggered by requests for maintenance that usually originate from clients. Requests may be related to various maintenance services, which trigger appropriate maintenance activities. Many factors influence timelines in maintenance request process. Timelines of maintenance request process significantly influence the quality of products and services offered to clients [11].

If there is a repository of various maintenance tasks conducted by a company, then it is possible to extract some information from historical data. Regarding the complexity of maintenance tasks, it is possible to extract tasks or alternatives that are most successful in terms of resource consumption. As there is no exact definition of the maintenance task complexity, the fuzzy screening
method is proposed for extraction of best alternatives for maintenance tasks. The set of best alternatives can be used to assess the needs of future maintenance task in terms of resources consumption.

The paper is organized as follows: Section II describes the basics of fuzzy screening method. Section III presents a case study based on real-life data sample, while section IV contains some conclusions and remarks.

II. FUZZY SCREENING

The fuzzy screening procedure is useful for the selection of a small subset of alternatives from much larger set of alternatives [12]. Each alternative is described by minimal information. The process involves multiple criteria attributes used for alternative description, as well as multiple experts whose opinion must be considered. Therefore, fuzzy screening procedure (FSP) is usually considered as a Multi Expert – Multi Criteria procedure.

The FSP consists of three components:
1. Set $X$ of $p$ alternatives $X = \{x_1, ..., x_p\}$.
2. Set $E$ of $r$ experts $E = \{e_1, ..., e_r\}$.
3. Set $C$ of $n$ criteria attributes $C = \{c_1, ..., c_n\}$.

Each expert for each alternative gives his/her opinion (score) how well that alternative satisfies each of the criteria. For this process, the scale $S$ of $m$ elements is used, where $m$ is usually 5 or 7. For $m=5$ scale $S$ is defined by: Very High (VH)-S5, High (H)-S4, Medium (M)-S3, Low (L)-S2, Very Low (VL)-S1. For $m=7$ scale $S$ is defined by: Outstanding (OU)-S7, Very High (VH)-S6, High (H)-S5, Medium (M)-S4, Low (L)-S3, Very Low (VL)-S2, None (N)-S1. Natural ordering applies to scale $S$: $S_1 > S_i$ if $i>1$, as well as the maximum and minimum of any two scores:

$$\max(S_i, S_j) = S_j \text{ if } S_i \geq S_j,$$

$$\max(S_i, S_j) = S_i \text{ if } S_j \leq S_i.$$ Very important is the definition of the negation for the scale $S$:

$$\neg(S_i) = S_{m-i+1} \quad (1)$$

In (1), $m$ is the number of scale elements.

For $m=5$ it is: $\neg(S_5) = S_1$, $\neg(S_4) = S_2$, $\neg(S_3) = S_3$, $\neg(S_2) = S_4$, $\neg(S_1) = S_5$. In other words:

$$\neg(VH) = VL, \quad \neg(H) = L, \quad \neg(M) = M, \quad \neg(L) = H, \quad \neg(VL) = VH.$$ Indepedently, each expert should provide an opinion on how important is each criteria attribute; this is done by using the same scale $S$. Vector $I=(q_1, ..., q_n)$ thus provides an importance of each criteria attribute for an expert. Each expert provides a separate vector of criteria attribute’s importance.

As each alternative is described by vector $V=(v_1, ..., v_n)$, where $n$ is the number of criteria attributes, and $v_j, j=1, ..., n$ are attribute’s values, the score $U$ for an alternative is calculated as follows:

$$U = \min_j [\neg(q_j) \lor v_j] \quad (2)$$

Example:

Let an alternative is described by values of four criteria attributes: $V=(M, L, H, VH)$, while importance of criteria attributes for some expert are defined by vector $I=(VL, M, L, VH)$. If the operator “$\lor$” is implemented as max, then the score $U$, calculated by (2) is:

$$U = \min (\neg(VL), M), \max(\neg(M), L), \max(\neg(L), H), \max(\neg(H), VH) = \min(\max(VH, M), \max(M, L), \max(H, H), \max(L, VH)) = \min(VH, M, VH, VH) = M.$$ As mentioned by Yager [12, 13], (2) can be seen as a measure of the degree to which an alternative satisfies the following proposition: “All important criteria are satisfied”.

After this stage of the process, each expert evaluates each alternative. If there are $r$ experts, then for each alternative is associated $r$ scores. Further process is an aggregation of scores in order to obtain overall score for each alternative. There are many aggregation operators, but in this case, Ordered Weighted Averaging operator (OWA) proposed by Yager in 1988 [13, 14] is used.

A. Aggregation function based on OWA operator

The next step in the process is definition of the aggregation function $Q$. As in [12] “This function can be seen as a generalization of the idea of how many experts it feels need to agree on an alternative for it to be acceptable to pass the screening process”. Besides some trivial, as well as more complex definitions of $Q$, the function which emulates the average is denoted as $Q(k)$:

$$Q(k) = S_{b(k)} \quad , \quad \text{where}$$

$$b(k) = \ln \left[ \frac{1}{r} \cdot \frac{k \cdot q - 1}{r} \right] \quad , \quad (3)$$

In (3) $\ln$ returns a value that is the closest to $a$, $q$ is number of points on the scale and $r$ is the number of experts.

An OWA operator of dimension $n$ is a mapping $F:R^n \rightarrow R$ that has associated vector: $w=(w_1, ..., w_n)^T$, where $w_l \in [0,1], 1 \leq l \leq n$. Furthermore:

$$\sum_{i=1}^{n} w_i = 1, \quad \text{and} \quad F(a_1, ..., a_n) = \sum_{j=1}^{n} a_j b_j ,$$

where $b_j$ is the $j$-th largest element of the bag $\langle a_1, ..., a_n \rangle$. After appropriately selected $Q$, it is possible to use the OWA method for aggregating experts’ opinions. The overall score $A$ for an alternative is calculated by (4):

$$A = \max_j (Q(j, r, B_j)) \quad (4)$$

In (4) $B_j$ is the worst of the $j$-th top scores.

Example:

Four experts evaluate an alternative so that the score of the first expert is $M$, the score of the second expert is $H$, the score of the third expert is $L$ and the score of the fourth expert is $VH$. By reordering these scores it becomes: $B_1=VH, B_2=H, B_3=M, B_4=L$.

For $q=5$ and $r=4$ according to (3) it is:

$$Q(1) = S_5, Q(2) = S_5, Q(3) = S_5, Q(4) = S_5$$

If the operator “$\lor$” is implemented as $\min$, the overall score $A$ of an alternative by (4) can be calculated as:

$$A = \max (\min(L, VH), \min(M, H), \min(H, M), \min(VH, L)) \quad , \quad A = \max(L, M, M, M) = M .$$
Therefore, the overall score $A$ for the alternative calculated as an aggregation of scores of each expert is Medium.

III. CASE STUDY

Case study is based on the real data extracted from the issues repository used in a very small software company with seven employees (classification according to [16]). Software development and maintenance activities are organized in the way that one or more programmers are assigned to each software application. When a maintenance request (MR) is received from a client, it is forwarded to a selected programmer from a set of assigned programmers. Each request may be solved by one programmer, or by a set of programmers assigned to a software application.

The first parameter included in the analysis is completion time ($\text{CompTime}$) for a request. This parameter is calculated as

$$\text{CompTime} = \text{ComplDate} - \text{RecDate}$$

where $\text{ComplDate}$ denotes the date when a programmer completes all tasks associated to the current request and enters that in the repository, while $\text{RecDate}$ denotes the date when the request is received from a software user and recorded into the internal application for tracking tasks.

Working hours spent on solving client requests provide the real basis for charging services to clients during software maintenance. These working hours are hours that a programmer spends on a specific task associated to a change request. In addition, these working hours could be also seen as a part of daily activities that programmers perform. Analysis includes the following types of working hours recorded in the repository:

- Working hours spent in the company ($\text{Company Working Hours}, \text{CWH}$). Recorded working hours that a programmer spends on activities in the company in order to solve assigned task related to change request. These activities are carried out using the internal resources of the software company.
- Working hours spent on Internet ($\text{Internet Working Hours}, \text{IWH}$). Recorded working hours that a programmer spends on activities that require Internet access to clients’ information system. These activities are related to direct access to software applications at client side.
- Working hours spent at client side ($\text{Client Side Working Hours}, \text{CSWH}$). Recorded working hours that a programmer spends at client side (in client’s company) in order to solve reported problem. These activities are performed when there is a need to get full insight into the situation that cause a problem or when a programmer should install or configure a software product.

Actually, completion time can be significantly greater than the sum of all tree types of working hours (see Fig. 1) because a task related to a request is usually scheduled in weekly plan of the assigned programmer.

A. Construction of data set

Data analysis is based on real data extracted from typical maintenance tasks recorded in the internal repository in the company. Each maintenance task is associated to one MR. Maintenance tasks included in the analysis are:

- Small modification of software (intervention on application user interface without affecting business logic and database layers),
- Minor enhancement (adding a small feature),
- Major enhancement (adding a more complex feature),
- Correction of an error (based on reported problem),
- Mandatory change (proposed by regulative),
- Providing support (responding to a client phone call).

For all selected tasks, analysis is based on the following, previously described, attributes: completion time, company working hours, internet working hours and client side working hours. However, for the analysis, three experts were used to assign discrete values to selected attributes for all six selected tasks. These discrete values are from the set $S$ containing VH, H, M, L and VL, as described in the second section of the paper. Assigned values are based on agreement between three experts that assessed attributes for selected tasks. Assigned values are completely based on subjective judgment of experts (which has been proved as reliable for experience experts [10]). Table 1 presents the discrete values assigned to the selected attributes for the set of six typical tasks. When judging about maintenance tasks, experts were forced to consider tasks complexity and the complexity of software

<table>
<thead>
<tr>
<th>Task</th>
<th>Completion time</th>
<th>CWH</th>
<th>IWH</th>
<th>CSWH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VH</td>
<td>H</td>
<td>M</td>
<td>VL</td>
</tr>
<tr>
<td>2</td>
<td>H</td>
<td>M</td>
<td>VL</td>
<td>L</td>
</tr>
<tr>
<td>3</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>VL</td>
</tr>
<tr>
<td>5</td>
<td>VL</td>
<td>VL</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>6</td>
<td>H</td>
<td>VH</td>
<td>L</td>
<td>M</td>
</tr>
</tbody>
</table>

Figure 1. Relation between maintenance request completion time and actual working hours
application affected by the task. The following remarks
should be considered while looking at the constructed data
set presented in table 1: a) Values for attributes present
relative experts’ judgments based on their experience and
considering the complexity of tasks extracted from
detailed requests descriptions (from the internal
repository), and b) The same value for completion time
for tasks 2 and 6 does not mean that the completion time
is the same, but rather it reflects the case of analyzing the
tasks with different complexity (larger completion time
for more complex task, and smaller completion time for a
simple task).

In addition, three experts were asked to assign values
from the set $S$ to assess the importance of attributes
selected for judging about the maintenance tasks. These
values are presented in table 2. Assigned values to
attributes’ importance indicate different attitudes of
experts towards the influence of attributes on maintenance
tasks. For the first expert, the most important is that he can
access the clients’ infrastructure via Internet, but he
prefers working in the company. The second expert thinks
that completion time is very important, and also the access
to clients’ infrastructure via Internet. The third expert
prefers working on client side or requires access via
Internet, but does not think that completion time is very
important.

<table>
<thead>
<tr>
<th>Expert</th>
<th>Completion time</th>
<th>CWH</th>
<th>IWH</th>
<th>CSWH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M</td>
<td>M</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>VH</td>
<td>M</td>
<td>VH</td>
<td>M</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>M</td>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

TABLE II.
ATTRIBUTE IMPORTANCE FOR SELECTED MAINTENANCE TASKS

B. Data analysis

Score for first alternative (see table 1), according to first
expert (see table 2) calculated by (2) is:

$$
\min(\max(M, VH), \max(M, H), \max(L, M), \max(H, VL)) = \min(VH, H, M, H) = M.
$$

Scores for each alternative (task) for each expert are
calculated analogously. Table 3 contains scores for each
alternative for each expert, as well as overall scores for
each alternative.

For $q=5$ and $r=3$ according to (3) it becomes:

$$
Q(1) = S_e = L, \quad Q(2) = S_e = H, \quad Q(3) = S_e = VH.
$$

Overall scores from table 3 are calculated by (4).

<table>
<thead>
<tr>
<th>Alternative (Task)</th>
<th>Expert 1</th>
<th>Expert 2</th>
<th>Expert 3</th>
<th>Overall score (OWA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M</td>
<td>M</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>2</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>3</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>5</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>6</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

The most satisfying alternative according to all experts
is the alternative one. This means that the most acceptable
option for maintenance task require optimal completion
time, and working hours spent in the company or
accessing client infrastructure via Internet.

IV. CONCLUSIONS

Maintenance tasks as the most costly part of the
software system life cycle can be evaluated according to
criteria attributes associated with every maintenance task.
The criteria attributes commonly refer to resource
consumption and they are related to task complexity.
However, task complexity is not exactly defined, and it
requires observing many technical, organizational and
social factors.

If there is a repository of various maintenance tasks
managed by a software company, then it is possible to
extract some information from historical data by fuzzy
screening procedure. As a final result of evaluation, the
prediction of future resources consumption for
maintenance tasks can be calculated.

Many promising directions for further work exist. The
first direction is related to inclusion of real data about
software complexity (number of code lines, number of
modules) and about maintenance staff skills (experience,
familiarity with software products and familiarity with
technologies) in the analysis. The next direction can be
related to testing the approach on public open source
repositories that are commonly used in empirical studies.
Comparing this approach with other commonly used
approaches on the same data sets is also possible research
direction. The most interesting direction will be
modification of the presented approach through
introduction of moderating parameters and comparison with
this approach.

ACKNOWLEDGMENT

Ministry of Education and Science, Republic of Serbia,
supports this research under the project “The development
of software tools for business process analysis and
improvement”, project number TR32044, 2011-2014.

REFERENCES

and evolution: a roadmap,” in Proceedings of the Conference on The
Future of Software Engineering (ICSE ’00), 2000, pp. 73–87.


M. Medin, S. Ebarasi, T. Fahlgren, S-E Johansson, S. Tornquist
and M. Holmgren, “Taxonomy of Problem Management Activities”, in Proceedings of the Fifth European Conference
on Software Maintenance and Reengineering (CSMR’ 01), 2001.

Software Maintenance Management. Information Technology and

Analysis Process to Characterize Software Maintenance Projects”, in Proceedings of the International Conference on Software
Maintenance (ICSM ’94), 1994, pp. 38–49.


