Benchmarking in Software Development

Chapter *in* IFIP Advances in Information and Communication Technology · January 1995

DOI: 10.1007/978-0-387-34847-6_20

20

Benchmarking in Software Development*

Nelly Maneva, Maya Daneva[a], Valia Petrova[b]

[a]Institute of Mathematics, Bulgarian Academy of Sciences
Acad. Bontchev str., bl.8, 1113 Sofia, Bulgaria
[b]Faculty of Mathematics and Informatics, University of Sofia
5, J.Baucher bul., 1126 Sofia, Bulgaria

1. INTRODUCTION

The Software Engineering solution of the so-called "software crisis" was proposed two decades ago. But there are still some alarming symptoms. A lot of software products are delivered with delay. They are of poor quality and with excessive costs (especially for maintenance). Though many methodological recommendations have been applied, our experience shows that pure theory itself is not enough for implementing the engineering approach to software development. So we propose a feasible approach based on common sense and pragmatism rather than on deep and complex theoretical results. In our opinion, people involved in software constructing are practically minded and such an utilitarian approach would be more comprehensible and useful.

2. A PROCEDURE FOR CONTINUOUS BENCHMARKING

The main goal of each software project is to ensure the efficient development of a high quality product. In order to achieve this goal we shall consider software project management as a sequence of crucial decision making which answers to the questions: What is the problem? What is the best alternative in given circumstances? What to do?

Usually decision making is done ad hoc and the motives of the resulting actions are vague. Next we are going to determine a unifying and systematic approach to supporting optimal decision making. This approach comprises two key ideas. First, the current objective must be posed (as Gilb write in [1]: "Actions without clear goal will not achieve their goals clearly"). Second, according to the objective stated, the kinds of objects and the set of their measurable attributes are chosen so as to benchmark them.

We suggest the following procedure for continuous improvement through benchmarking:

*This research is partially supported under the contract I-24 with the Bulgarian Ministry of Education and Science.

A. Rolstadås (ed.), *Benchmarking — Theory and Practice*

1. State the goal.
2. Define the relevant objects and their attributes.
3. Define the set of competitive objects and compare them.
4. Plan and accomplish the appropriate course of actions.
5. Measure results against goal.
6. Re-evaluate and continue.

The problems of applying this procedure to a software development and use are:

a) to submit a scheme of object description, a method of object comparison and a software tool supporting that method.

b) to identify the situations at selected moments of the Software Life Cycle which the procedure is used for.

Next we are going to propose a solution of the problems just mentioned.

3. OBJECT DESCRIPTION AND COMPARISON

Further on we shall stick to the following definition given in [2]: "*An abstraction of an object is a characterization of the object by a subset of its attributes. If the attribute subset captures "essential" attributes of the object, then the users need not be concerned with the object itself but only with the abstract attributes*".

A scheme of an object description and comparison based on the above definition will be given below.

Let S be a set of homogeneous objects $S = \{S_1,S_2,..,S_n\}$ and $H = \{H_1,H_2,..,H_m\}$ - a set of m attributes which describe the objects of S.

Each element S_i of S can be presented by an ordered m-tuple:

$$S_i = (E_1^i,...,E_k^i,..., E_m^i),$$

where E_k^i is the evaluation rating of the i-th object S_i with respect to the k-th attribute H_k.

The decision what elements should be included in sets S and H depends on the predefined objective. The latter can be related to a certain decision making problem. Each attribute must be weighted in accordance to its relative importance.

The heuristics algorithm [3] accomplished over descriptions is suggested to be used for transforming the set S into the set S', where the set S' is a completely ordered list of examined objects. This algorithm ranks the objects with respect to their capability of supporting the defined objective .

The object description and comparison can be performed by means of the software system SSS [4]. The latter was primarily designed as a tool for software product selection, but now being slightly modified it can be used for arbitrary objects. SSS comprises the following components:

The *Object Manipulating component* presents objects through tables where rows correspond to objects and columns - to attributes. The component ensures table creation, table deletion, data input in a table and a table structure modification through adding or deleting objects/attributes.

The *Weights Defining component* supports different modes of assigning the weights to the set of attributes.

The *Ranking and Result Presentation component* ensures the establishment of some parameters of the heuristic algorithm and its accomplishment. Next the component presents the ranked objects in different forms - textual or graphical.

4. HOW TO CONTROL SOFTWARE DEVELOPMENT THROUGH BENCHMARKING

The proposed procedure for continuous improvement can be applied in different situations. Each situation may be described as follows:

TO *<activity>*
FROM THE VIEWPOINT OF *<kind of software personnel>*
TARGET OBJECT *<object to be studied>*
SO AS TO *<objective>*

The item *<activity>* can be estimate, predict, choose, assess, describe, evaluate, etc. But all of them can be grouped in two main benchmarking activities:

a) *ANALYSIS* implies the comparison of a target object against the preliminary established model. The model is an abstract representation of the object with artificially constructed attribute values which must be achieved or avoided. So the model describes the success or failure and can be used for predicting the effect of some actions on the product or on the process.

This activity involves the describing of a "standard" object which will serve as a benchmark. Usually this requires a thoughtful study of the object and the use of some predicting procedures for determining unknown parameters and interpreting the results.

b) *ASSESSMENT* implies the study of a set of existing objects and their ranking so as to obtain the information needed for decision making, i.e. to select the most appropriate alternative for solving the problem under consideration, to see the position of a particular object among its competitors, etc.

The benchmarking can support the decisions made by each participant in software development and use, e.g. the item *<kind of software personnel>* can be managers, marketing staff, software or process engineers, programmers, vendors, users, etc. Their different points of view determine the diversity of objectives and the variety of target objects.

Modifying the classification given in [5], we can group the *<object to be studied>* as follows:

- Products - any artefacts, deliverables or documents which are created during the software development (specification, designs, programs, test data, reports, etc.);
- Processes - any software related activities (design, programming, testing, reviewing, auditing, etc.);
- Resources - i.e. personnel, teams, hardware, software, offices;
- Others - service policies, training programs, etc.

The *<objective>* stated depends on the problem under consideration.

There are no restrictions for the studied objects. If one can construct a set of measurable attributes and evaluate them, then the proposed procedure will work. But sometimes choosing the right target object and its attributes is not a simple task [5]. It requires joint efforts of experts, who keep track of evaluation practice and can advise when, what and how benchmarking should be carried out.

In case of complex target object a hierarchical characteristics structure can be used so as to define the measurable attributes. For example, the development process is such an object. It comprises a set of activities performed during the software development, the scheduling of these activities and the manipulating of the product. The development process is presented as shown below:

```
                         process
                        /       \
         process_technologies          process_performance (as a project)
       /                    \
   development_technologies  SQA_technology
   /       |        \
 design   coding    integration ...
```

A development technology specifies the methods and procedures used during a particular phase of development. A SQA technology determines planning and control procedures referring to a certain software product and process. Each SQA technology includes all aspects of the process discipline, i.e. documentation, standards and organization rules which have to be followed.

The above described stepwise decomposition determines when the process decision making takes place. The SQA technology must be chosen at starting the software project while the development technologies depend on the current Software Life Cycle phase and they are determined in the process. The two types of technologies should be modified according to environment changes. At such moments the proposed benchmarking procedure will support the decision maker and it will help him to choose the optimal solution.

5. SOME EXAMPLES AND EXPERIMENTAL RESULTS

We have briefly described some specific situations of decision making through a Software Life Cycle.

5.1. Analysis phase

CASE 1

TO assess

FROM THE VIEWPOINT OF users

TARGET OBJECT software product of a chosen (given) type

SO AS TO get some information about users' attitudes towards such products

Attributes: Some users' defined quality characteristics.

Set of Objects: All software products available at the market.

EXAMPLE: The class of Illustration packages is studied by using some data from [6]. The attributes together with the user defined weights are given in Table 1.

The SSS system has been applied twice: with equal attribute weights and with the weights, presented in [6]. The results of ranking are shown in Table 2.

Table 1

Attributes:	Weights:
al - Quality of Output	4.35
a2 - Ease of Use,	4.25
a3 - Firm's Reputation	4.12
a4 - Value	4.12
a5 - Charting Capabilities	3.89
a6 - Presentation Features	3.88
a7 - Drawing Tools	3.63
a8 - Service and Support	3.47
a9 - Spreadsheet links	3.32
a10 - Price	3.27

Table 2

Software Products	Ranking I	Ranking II
Harvard Graphics	2	1
Freelance Plus	3	2
PowerPoint MS	1	3

5.2. Feasibility phase

CASE 2

TO select
FROM THE VIEWPOINT OF managing staff
TARGET OBJECT virtual project
SO AS TO select the optimal virtual project on the basis of its economic, technical and market feasibility.

Attributes: Potential Sales Volume, Level of Competition, Compatibility with Marketing, Compatibility with Production, Patent Protection, Similarity to Existing Products, Environmental Compatibility.

Set of Objects: All virtual projects presented.

CASE 3

TO analyze
FROM THE VIEWPOINT OF project leader or process engineer
TARGET OBJECT technology
SO AS TO benchmark it

A typical task at this phase is to evaluate the production environment in a certain software firm. The process engineer has to establish the model of the desired project technology and has to evaluate it against the current technology used in the firm.

Attributes: The measurable attributes can be defined after the decomposing the following criteria:

a) Functional criteria which represent the quality of a technology: modularity, integrity, clarity of the methods used, precision, effectiveness, level of complexity;

b) Performance criteria which represent the quality of a current technology in use: adaptability, flexibility, level of automated support, level of standardization, reliability, efficiency, productivity;

c) Organizational criteria which represent the quality of the technology discipline: the abilities of being controlled and coordinated, level of communication complexity (among groups and within groups), management complexity.

5.3. Design phase

CASE 4

TO select
FROM THE VIEWPOINT OF user or project leader
TARGET OBJECT information technology
SO AS TO determine the most appropriate information technologies for the software development.

Attributes: Ease of learn, Power, Efficiency, Program Volume, Structure, Portability.

Set of Objects: All information technologies available.

EXAMPLE: Let us compare a number of Programming Languages. The attributes used and their expert defined weights are given in Table 3.

Table 3

Attributes:	Weights:	Attributes:	Weights:
al - Ease of Learn	3	a4 - Programs Volume	6
a2 - Power	9	a5 - Structure	7
a3 - Efficiency	5	a6 - Portability	8

Two cases have been studied - with equal weights and with weights, defined by means of the SSS system.

The attribute values and the ranking related with the studied cases are given in Table 4.

Table 4

Programming Languages:	Ranking I:	Ranking II:
COBOL	4	5
FORTRAN	6	3
BASIC	2	2
PL/1	7	6
FOC	5	7
RPG II.	3	4
ADA	1	1

CASE 5

TO compare

FROM THE VIEWPOINT OF project leader or process engineer

TARGET OBJECT corrective actions

SO AS TO improve the process of designing

Attributes: process parameters improved by the performed corrective actions i.e. design quality, project cost and resources, degree of process control, etc.

Set of objects: Possible corrective actions for concerning the design process, e.g.:

- checking the guidelines followed;
- clarifying the design guidelines;
- re-organizing the design group;
- adopting the design standards.

5.4. Programming phase

CASE 6

TO compare

FROM THE VIEWPOINT OF quality engineer

TARGET OBJECT project state

SO AS TO control the quality during the development of a new software product

Attributes: Reliability, Authorization, File Integrity, Audit Trail, Continuity of Processing, Service Level, Access Control, Methodology, Correctness, Ease of Use, Maintainability, Portability, Coupling, Performance, Ease of Operation.

Set of Objects: Consequence of project states.

CASE 7

TO select

FROM THE VIEWPOINT OF designer

TARGET OBJECT integration strategy

SO AS TO find out the best integration strategy to be applied for constructing the software system out of the program units.

Attributes: Partial Integration, Time Needed to Construct a Working Program Version, Use of Drivers, Use of Dummy Section, Parallelism, Ability to Test, Program Paths, Ability for Controlled Testing, Inefficiency.

Set of Objects: All available strategies.

5.5. Evaluation phase

CASE 8

TO assess

FROM THE VIEWPOINT OF project leader or user

TARGET OBJECT program documentation

SO AS TO rank the software product documentation according to its quality

Attributes: Style, Correctness, Completeness, Structureness, Clarity, Compliance with Standards, Useful Examples, On-line Help.

Set of objects: Documentation of the competitive software products.

CASE 9

TO assess

FROM THE VIEWPOINT OF project leader

TARGET OBJECT participants in a software project

SO AS TO establish each participant's contribution to the software project progress.

Attributes: Productivity, Planned Participation in the Work on the Project, Real Participation in the Work, Balance Based on Planned and Real Results, Quality of Results.

Set of objects: Participants can be divided into three groups: management staff (project leader), specialists (designers, programmers) and administrative/service staff (technicians).

5.6. Use.

CASE 10

TO evaluate

FROM THE VIEWPOINT OF user

TARGET OBJECT software product

SO AS TO establish the position of the new software product among the products at the market.

Attributes: Correctness, Reliability, Efficiency, Integrity, Usability, Portability, Reusability, Interoperatability, Testability, Flexibility, Maintainability.

Set of Objects: All products from a given class.

CASE 11

TO evaluate

FROM THE VIEWPOINT OF user

TARGET OBJECT software service

SO AS TO establish the quality of service provided by the competitors with respect to a certain software type.

Attributes: Context-sensitive help, Unlimited free support, Toll-free support, Daily Support, Weekend Support, BBS Support, Fax Support, Other Extra-cost Training or Support.

Set of objects: Software products from a given type.

EXAMPLE: Using data from [6] the set of service policies provided for a number of Graphical Packages is studied. The attributes with the mentioned weights are given in Table 5.

The SSS system has been applied twice - with equal attribute weights and with the weights, presented in [6]. The attribute values and the results of ranking are shown in Table 5 and Table 6 corresponding.

Table 5

Attributes:	Weights
a1 - Context-sensitive help	8
a2 - Unlimited free support	5
a3 - Toll-free support	4
a4 - Daily Support	9
a5 - Weekend Support	3
a6 - BBS Support	6
a7 - Fax Support	4
a8 - Other Extra-cost Training or Support	3

Table 6

Objects:	Ranking I	Ranking II
Aldus Persuasion 2.1.	5	5
Charisma 2.1	1	1
Freelance Graphics	3	4
Harvard Graphics	2	2
Hollywood 1.0v2	6	6
PowerPoint 3.0 MS	4	3

CASE 12.

TO evaluate
FROM THE VIEWPOINT OF user
TARGET OBJECT training programme
SO AS TO compare the quality of training offered by the competitors and to decide which programme aspects have to be modified in order to ensure more efficient training.

Attributes: Number of computers used for Training, Training Time, Number of Participants, Number of Training Units, Place of Training (a Training Center or Firm's office).

Set of objects: Training programmes available.

6. CONCLUSIONS

The paper describes a feasible approach to benchmarking for software development and use.

Our further research will be focused on:

- defining or precising (if chosen) a set of all reasonable attributes for some objects often used in software development. Next the person dealing with benchmarking is supposed to select an appropriate subset of the attributes thus defined;

- designing and implementing a prototype of an intelligent system facilitating object description and ranking on the basis of different methods.

REFERENCES

1. T. Gilb, Principles of Software Engineering Management, Addison Wesley, 1987.
2. P. Wegner, Programming languages- the first 25 years, IEEE Trans. on Computers, C-25, 12 (1976).
3. E. Anderson, A Heuristic for Software Evaluation and Selection, Software Practice and Experience, 8 (1989) 707.
4. M.J. Daneva and N.M. Maneva, A Software Selection System - Description and Applications, Proc. of ACMBUL Conference "Computer Applications", Bulgaria, Oct. 4-10, (1992) 26-1.
5. N. Fenton, Software Metrics: Rigorous Approach, Chapman&Hall, 1991.
6. C. White, Harvard Still Tops in Graphics, PC WORLD, 11 (1992) 228.