Software Design Tools Evaluation in the Context of a Metaparadigm

Peter Kokol, Bruno Stiglic, Viljem Zumer

University Maribor, Faculty of Technical Sciences, Smetanova 17, 62000 Maribor, Slovenia

Abstract. The idiosyncrasy between rapid advancements in the information technology and the unsuccessfulness of many software systems was the reason for construction of a new software design approach called metaparadigm. Metaparadigm is a paradigm for designing software system design paradigms and was used for the development of a new evaluation paradigm presented in this paper.

1. Introduction

Software system design is currently delineated by the curious situation in which on the one side we observe the rapid advancements in the information technology and on the other the unsuccessfulness of many software systems. This idiosyncrasy between possibilities and results was the reason for performing a study in the manner to encounter possible sources of inadequate design and a feasible solution for them. The results of the research were:

A: software system design:

1. is the implementation of a human activity system [1] (a human activity system is a purpose system which expresses some purposeful human activity which could be in principle found in the real world. Human activity system is nominal in the sense that they are not description of real world activities, but are intellectual constructs);

2. is a process performed in a specific design situation, according to a specific design paradigm;

3. is a goal-oriented decision making exploration and learning activity;

4. occurs in two contexts: the context within which the designer operates and the context produced by the designing design itself;

B: relations between software design situations, processes and paradigms show that:

- there are no absolute good or bad design paradigms, they perform good or bad only in specific design situations;

- there are design situations in which no known design paradigm can perform successful.

According to above findings we contended that to enable successful software system design in any design situation we must first design an appropriate design paradigm. This activity was called the metadesign and a metadesign paradigm a metaparadigm [10]. An adequate metaparadigm should enable to design, to describe and to evaluate design paradigms. In this paper we will concentrate on the last topic, especially on the case where software design is accomplished by end users using software design tools.

Although these tools have enormously eased the burden of designing software systems for untrained end users, they have also created a confusing, difficult problem in tools evaluation and selection because of [1, 6, 15, 16]:

1. the huge number of available software design tools;

2. the rapid technological change in the software technology;

3. the lack of appropriate technical expertise of end users;

4. the lack of applicable evaluation paradigms.

The Scope of the Paper

While much has been written about software evaluation in general the guidances and recommendations previously provided are to general to be of practical use. Indeed, conventional evaluation paradigms are to specific, incomplete and to narrow [1, 6, 14, 15] There after
in a context of a metaparadigm and using it we have decided to develop a more adequate evaluation paradigm. The main advantages of our paradigm are:

- a solid theoretical framework;
- specific guidances enabling the user to evaluate a system for every specific application (i.e. business, health-care, administration), dimension (i.e. usability, functionality, reliability, portability) and level (i.e. paradigm, methodology, tool);
- strict and well defined taxonomic space;
- fair ranking approach;

In the present paper we will therefore:

- introduce a metaparadigm and its use as a vehicle for designing evaluation paradigms;
- present a new evaluation paradigm;
- show the applicability of the new evaluation paradigm in selection of software design tools according to their usability.

2. Metaparadigm

Before describing the metaparadigm in more details we would like first to define the idiom paradigm. According to some basic denotations stating that the paradigm is (1) a set of universally recognized scientific achievements that for a time provide a model of solutions to the community of practitioners or (2) a set of meta-theoretical assumptions about the nature of the subject of study, we have defined it as a pattern for executing actual actions in the real world.

Using the last definition we claim that the

metaparadigm is a pattern for executing actual design actions in designing real world-design paradigms.

An adequate metaparadigm should permit the following minimal set of activities:

- formal description of design paradigms, design process and design situations;
- formal evaluation and comparison of design paradigms;
- metadesign of design paradigms, where metadesign is treated in a very broad sense as invention of new design paradigms, adoption of known design paradigms, composition of known design paradigms, selection between design paradigms etc.; and
- learning, accumulation and reuse of knowledge.

The metaparadigm and its use in software system design is shown in Fig. 1.

Emphasizing above arguments and respecting recent scientific findings (the crisis of the traditional scientific approach [3], tendency toward more softer system thinking [3], research paradox [10] and the software crisis [13]) we decided that a metaparadigm should be pluralistic, useful, systemic, easy to use and recursively adaptable (recursive adaptability states that a metaparadigm must be an iterative learning system accumulating new knowledge about metadesign, software system design and related).

The metaparadigm should be composed out of a suitable

- framework,
- theory and
- metadesign methodology and philosophy.

In our first attempt we have selected the idea of a process as the framework, the process formalization (metamodeling) as the theory and the Checklands Soft System Methodology (CSSM) as the methodology and philosophy. Some important arguments for the selection were:

- the applicability of CSSM for researching human activity systems,
- the similarity between definitions of a process and a human activity system,
- the strong relationship between a process and its
formalization.

The metaparadigm and its components are described in more detail elsewhere [3, 10, 11, 12], so we will briefly describe just some for the understanding essential points of the Checklands soft system methodology in the next subsection.

The metaparadigm and its components are described in more detail elsewhere [3, 10, 11, 12], so we will briefly describe just some for the understanding essential points of the Checklands soft system methodology in the next subsection.

Figure 1. The structure and use of the metaparadigm in designing software systems

2.1. Checklands soft system methodology (CSSM)

Following recent research which shows that the conventional scientific approach is not successful in solving unstructured and complex real-world problems, we have selected the CSSM [3] as the appropriate paradigm design methodology. The CSSM uses system ideas to formulate basic mental acts of four kinds: perceiving, predicting, comparing and deciding on actions. The CSSM begins with a soft and unstructured expression of a problem situation, and continues with the definition of some human activity systems which seems relevant to the problem situation, named root definitions. Every root definition should include six crucial characteristics namely Customer, Actor, Transformation, Weltanschauung, Owner and Environment (CATWOE elements). These definitions are used next as the basis for making conceptual models of systems being selected.

After the model is built it is compared with the situation perceived. The primary aim of the comparison is to comprise a debate, discussion or argument out of which suitable changes can be made. Once the debate stages have revealed possible changes, then the new problem situation becomes that of implementing these changes in the real world. The CSSM never solves problems out of existence, it is just the never ending learning process which should possibly improve the problem situation, and enables with the gained new knowledge to start another cycle of learning.

2.2. Metaparadigm and evaluation paradigm design

The evaluation of software design paradigms, software design processes, software design tools etc. is an integral part of a software design paradigm and can be thereafter designed using the metaparadigm too. (see Fig. 2.)
3. The new evaluation paradigm

As the kernel for root definition of our new evaluation paradigm we have taken the classical systems theoretic definition [2] stating that

to evaluate a system is to measure its parameters in terms of well-known metrics using standard instruments so as to analyze it against generally accepted benchmarks.

In that manner we have defined the evaluation as follows:

evaluation is the process in which for every specific application, specific taxonomic space, instruments and ranking approach are defined first and then used to rank IS design paradigms. The process considers achievements from other scientific fields and enables the recording and reuse of obtained knowledge.

The conceptual model of above root definition consist out of construction, ranking and knowledge subsystems (Fig. 3.). The construction subsystem uses knowledge obtained by the knowledge subsystem to define taxonomic space, instruments and ranking approach, transferring them to the ranking subsystem and the new knowledge to the knowledge subsystem. The ranking subsystem uses the taxonomic space, instruments and the ranking approach in the transformation of an unordered set of IS design paradigms to an ordered one. The knowledge subsystem acquires and records external and internal knowledge.

4. The evaluation of software design tools usability

In this section we address the usability evaluation of microcomputer software design tools from the possible perspective of theirs end users (the advantages and weaknesses of end-user computing have been in length discussed in software literature [i.e. 13, 19, 20]). We can state that to date usability has not been addressed comprehensively in the software literature excepting a paper from Lowery and Martin [16]. They have gathered pertinent papers from medical and general software literature on software usability and evaluation and introduced a new framework for the conceptualization of usability which was used as the basis for the construction of our usability evaluation paradigm (UEP). However we had have to improve the evaluation criteria and to define more precise metric units. According to Lowerys framework the usability is defined [16] as

the capability to be used by humans
easily and effectively and should be evaluated from six different dimensions:

- **logical organization of procedures**: the procedures performing a task within a system should follow a logical organization that is meaningful to the user. A logical organization helps to understand and predict the system behavior, which not only reduces the time to learn and use the system but also increases the user satisfaction. In the case of application software (software developed for specific applications in specific organization is denoted as application software in this paper) this dimension can be evaluated by comparing the old manual procedures with the new automated ones. This can be done in informal manner by interviewing users or in more formal one by using the theory of graphs [21]. Formally the logical organization can be evaluated using the graph theory seeing the organization as the hierarchical tree, acyclic or cyclic graph, the graph with one-way connection etc.;

- **screen design for data entry**: the presentation of information on the screen of the user's video display terminal plays a significant role in the usability of a system. This dimension can be evaluated formally by using results from cognitive sciences like gestalt laws, information content, information flow etc. [17, 21] or simply by the versatility and ease of use of screen design functions;

- **error handling**: the detection and reporting of errors is an important dimension in effective and efficient use of a software system by an inexperienced user, which can be evaluated according to the information content of the error message;

- **data retrieval and report generation**: reporting is of the particular interest to the user. He should first retrieve data in an easy but effective way and than report them in a suitable form. This dimension can be evaluated according to the number and ease of use of retrieval and report generation functions. For example they can be accessed through mnemonic or natural language interface; the report generation functions can enable editing of reports or enable the generation just of standard reports; the retrieval function set can include just basic functions (Join, Select, Display) or also mathematical and statistical functions like Count, Average, Minimal, etc. In the case of application software we can evaluate the generated reports by comparing them to old manual written ones;

- **learning/help aids**: learning and help aids provide instructions on how to use a system, including instructions for such activities as data entry and retrieval, using menus, opening and closing of databases etc. This dimension can be evaluated according to the class of help aids i.e. static written documentation, on line help, context-sensitive help, and the

![Figure 3. The conceptual model of our new evaluation paradigm](image-url)
quality of written documentation (i.e. ease of read, good examples, cross references, indexes) etc [11];

- consistency: the system should be as consistent as possible across all tasks and modules. Consistency enables the user to make a prediction of system behavior, which in turn helps him to learn, understand and accomplish a given task in an effective manner. Consistency can be evaluated according to consistency of function keys, menu item selections, break keys, escape keys, error messages, command syntax etc.

In the Table 1, we present a possible taxonomic space consisting of above six dimensions. It is clear that this space represent just a general view to the usability evaluation problem. For more specific applications we can define different measures, divide the dimensions in subdimensions, select other evaluation criteria and so on.

<table>
<thead>
<tr>
<th>DIMENSION</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORGANIZATION(^1)</td>
<td>TR</td>
<td>AG</td>
<td>TAG</td>
<td>CG</td>
<td>TCG</td>
</tr>
<tr>
<td>DATA ENTRY(^2)</td>
<td>NF&amp;CD</td>
<td>NF&amp;WD</td>
<td>MNC</td>
<td>MC</td>
<td>NLC</td>
</tr>
<tr>
<td>ERROR HANDLING</td>
<td>no</td>
<td>single</td>
<td>type of</td>
<td>suggest</td>
<td>repairs</td>
</tr>
<tr>
<td>RETRIEVAL AND GENERATION(^2)</td>
<td>SF, SR, MI</td>
<td>SF, ER, MI</td>
<td>CF, SR, MI</td>
<td>CF, ER, MI</td>
<td>CF, ER, MI</td>
</tr>
<tr>
<td>LEARNING/ HELP(^5)</td>
<td>BQ, NH</td>
<td>GQ, NH</td>
<td>BQ, OH</td>
<td>BQ, CSF</td>
<td>GQ, CSF</td>
</tr>
<tr>
<td>CONSISTENCY(^5)</td>
<td>G-, M-, C-</td>
<td>G+, M-, C-</td>
<td>G-, M+, C-</td>
<td>G+, M+, C-</td>
<td></td>
</tr>
</tbody>
</table>

1 TR - tree structure organization, AG - one way acyclic graph, TAG - two way acyclic graph, CG - one way cyclic graph, TCG - two way cyclic graph

2 NF&CD - no screen design functions and standard entry is clumsy, NF&WD - no screen design functions and standard entry is well designed, MNC - mnemonic screen design commands, MC - menu oriented screen design commands, NLC - natural language screen design commands

3 SF - standard relation functions, CF - mathematical and relation functions, MI - mnemonic interface, NI - natural language interface, SR - standard report, ER - enables report editing

4 BQ - no or bad quality documentation, GQ - good quality documentation, NH - no HELP, OH - on-line HELP, CSF - context sensitive help

5 G+ consistency of global keys (HELP, BREAK, ESC ...), M+ consistency of menu keys (ENTER, EXIT, PgUp, Down, Up ...), C+ consistency of command syntax

4.1. The Usability Evaluation Process

The usability evaluation process has been derived from the conceptual model described in the previous section and is defined in an informal manner with a simple natural language metamodel as the following algorithm:

1. Using empirical measurement obtained by interviewing appropriate experts, place the desired software systems into the taxonomic space.
2. Repeat step 1 until all systems are properly positioned.

3. Use the Andersons heuristic [1] to rank the paradigms using measures obtained in step 1.

4. Acquire and record the obtained knowledge.

4.2. An Example

Above process and the taxonomic space from Table 1 have been used to evaluate the usability of four popular software design tools from the end-user perspective. They were selected first on the basis of theirs popularity in our country and second because they are representatives of four most to end-user computing applicable groups, namely:

- **Spreadsheet software**: Lotus 1-2-3 V3.1;
- **Natural language interface**: Lotus 1-2-3 with Hall V3.1;
- **Integrated package**: Symphony V3.0;
- **Query language**: DBase 3+.

Resulting measurements obtained by interviewing appropriate experts and the ranking are shown in Table 2. We see that Lotus with a natural language interface is the most usable tool among tools selected according using our evaluation paradigm.

Table 2. The rankings of four popular software packages

<table>
<thead>
<tr>
<th>TOOL</th>
<th>ORGAN.</th>
<th>ENTRY</th>
<th>ERRORS</th>
<th>RET&amp;GEN</th>
<th>LEARN.</th>
<th>CONST.</th>
<th>RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEIGHT</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>LOTUS</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>LOTUS+HAL</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>SYMPHONY</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>DBase</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

5. Conclusion

We have introduced a metaparadigm and a new evaluation paradigm in this paper. It is our belief that this paradigm overcomes some major weaknesses of older and more conventional ones because of following advantages:

- a solid theoretical framework;
- specific guidances;
- a strict and well defined taxonomic space; and
- a fair ranking approach;

Although we have presented only a specific applications of our new paradigm, we belief that it is appropriate also for evaluation of other dimensions of software design tools like reliability, functionality, portability etc.

6. References


