The environment of cognitive-contextual software design: from concept to implementation

A I Razumowsky
Trapeznikov Institute of Control Sciences RAS, 65, Profsoyuznaya street, Moscow 117997, Russia

E-mail: razumowsky@yandex.ru

Abstract. Two main points of the objectives to design software systems (SS) have been considered: achieving proper functioning of the SS with consistent results, and the possibility to obtain reliable assessments of the Software System quality depending on the initial requirements. A cognitive-contextual design strategy has been proposed that can significantly improve the perception of actual elements of the design task by creating a context with selected formalized conditions, and enhance cognitive coherence of the project elements throughout the Software System lifecycle.

1. Introduction
A criterion of any activity is its practice, and the practice should demonstrate significant results. The practical focus of Software System design (SS) should be based on the one hand, on achieving their proper functioning with consistent results, and on the other, on the possibility to obtain reliable quality assessments of the finished product depending on the initial requirements. This necessary balance can only be observed and maintained when creating coordination and synchronization between cognitive capabilities of information and its abstract representations within the design environment.

The process of creating Software Systems depends closely on the influence the human factor has on the environment. The Software System is often superficial and misused. It is believed, for example, that firstly, the design process has a completion point in the Software System lifecycle, and secondly, it is trivially formalized and thus can be measured and calculated. The belief that any Software System project can be completed is easily found by the well-known cliches of this kind: "analysis and development at early stages of design". This means that the entire lifecycle activity is conditioned on the analysis at "early stages" of the project. In other words, it is assumed that at the stage of task-setting and analysis, it is possible to realistically identify the main attributes of a particular Software System including estimation of the time and cost of work, the amount of human resources and the factors of efficiency and optimization of the finished product. Such an approach would not be beneficial. On the contrary, modern Software System design methodologists write differently: "Only when you have finished your work, you have a full knowledge of how it should have been done" [1].

The conceptual level of task-setting and representation of an architecture of its solution is individual and in line with minimization of the time to access any arbitrary information element (IE). That is, the information and structural outline of the project should be formed based on the following postulates:
• Information Elements are homogeneous in their cognitive perception (which will ensure equal time of attention from the designer to each such element);
• Information Elements are homogeneous in terms of coordinates, in other words, they are equivalent to a single common priority by the way of how they are stored;
• each task contains a unique set of Information Elements;
• each problem is solved individually depending on the designer.

2. Methodology

The main factor in building the concept is the following information attributes - individual reputation, recognizability and utilization of the project elements. For example, the use of design pattern (DP) is only possible if you have proper experience with specific and individually known patterns, while the use of obscure and unclear patterns is difficult to construct an effective concept. In addition, atomic data may at any time need consideration to gain their better understanding. Consequently, information storage, visualization and perception is the necessary basis to build a concept, representing a kind of supporting attributes of the design context. Let us note that, the building of a concept does not involve practical templates for a particular solution to a task as opposed to analysis (or analysis at early stages of the project). It refers to creating special conditions and comfort of design where the foundation will be individually known and utilized elements and their compositions. Then we can expect more confidence in decision-making, and greater flexibility in their correction.

So, let’s define the term "concept" of the project as the establishment of a unique information context, that is, a special order of the Information Element location among an arbitrary set of initial and secondary data of the lifecycle (LC) of the Software System, which is quite convenient for perception, assessment and decision-making.

The concept aims to create a comfortable, substantial and descriptive context to implement principal activities. Contextual environment designed conceptually can combine in a single perception field the elements of the solved task, both formal and not transformed into a formal or semi-formal appearance, namely, their original content expressed in a comfortable and individually recognizable form. In addition, the concept of context arrangement will combine both analytical and synthetic – that is, effective and constructive - characteristics and solutions. Thus, the concept-building phase directly paves the way for implementation processes through writing a program code, debugging or building testing schemes.

It should be emphasized that the conditions and capabilities of the contextual design environment formed in the concept creation not only determine the procedure for and quality of the Software System implementation but also increase the susceptibility to any change in decisions. Changes in the decisions taken will be facilitated, firstly, by the accumulation in the contextual environment of secondary data occurring at all stages of the Lifecycle, which are, for example, the results of the Software System performance and testing, and secondly, by the dual presentation [2] of the Information Elements in a single context.

The practice of dual presentation of information in terms of cognitive perception is not well studied today. Among the small exceptions, there is the work [3] that studied effects of interaction between system concepts useful for users and perceived simultaneously with comfort and usefulness in practice, and the possibility of design transformations after the system stopped to be used. The individual perception of data in building systems was investigated in the work [4]. The difference in the perception of information in designing and testing a program code is discussed in the work [5]. The work [6] is devoted to the review of how a human personality influences decision-making subject to the context of software design. The work [7] proposes a strategy for modeling cognitive context within the conventionally true semantics. The authors suggest that this pattern of the prediction creates a concept of conventional true value where cognitive context captures a set of lexical limitations and forced modifications.

When all conceptually significant Information Elements are gathered in a single context, their interaction with each other becomes clear, which is required to make more objective assessments and
take informed decisions. Therefore, the formation of the concept is the creation of an environment of individual cognitive comfort of activity in terms of the designer’s individuality and individuality of the task data. The higher the comfort is, that is, the freedom of perception and decision-making, the more efficiency should be expected from a particular designer in the implementation of a particular project.

The main focus of the information model of contextual-cognitive design (figure 1) is to develop individual experience as a fact that determines the possibility of cognitively coherent implementation of the project. It is important to underline that the Software System quality may be improved throughout its Lifecycle only under conditions when the corresponding design elements are perceived as a whole. Any cognitive integrity damage entails discomfort in making right decisions and, as a consequence, the loss of the Software System quality.

![Figure 1. Contextual cognitive design information model.](image)

Experience formed through successive iterations of accumulation and perception of the information set of the project, awareness of the significance of each of its elements, attempts to combine and associate Information Elements with each other and as a whole, finally gives impetus to implementation or, on the contrary, if knowledge (or experience) is insufficient, it prevents the manifestation of decisions.

The information model for implementing the Software System (figure 2) determines the sequence of the developed cognitive coherence of the project which depends, on the one hand, on the selection and quality of perception of the Information Elements, and, on the other hand, on the dynamics of results formation, together with which the initial Information Elements are preparing the ground for the next iteration of the software design. The first step in the implementation is the onset of intuitive readiness for implementation when the perception of the existing Information Element complex is sufficient to understand the initial ways of design solutions.

Then, comes the moment of possibility of generalization or, in other words, conditions are established to combine the Information Elements in generalizing unifications (GU), namely, groups, data structures, algorithmic sequences, classes, patterns, etc. It should be noted that the Generalizing Unification production stage is difficult without proper utilization of the Information Elements, least
because it is impossible to generally rely on superficial or obscure knowledge about the Information Elements. Even worse, this moment will come again and again at each subsequent iteration of design with ever less effective decision-making. On the contrary, when perceiving well-recognized, utilized and individually familiar Information Elements, a moment of cognitive coherence of the project will come. This is a landmark situation that controls the cognitive integrity of the project or a local solution. Such control cannot be overestimated because it creates a sense of satisfaction with the designer's work and prevents blind actions. The next two steps correspond to the development of functionality for the final result, through which the transition is made to the next iteration of the software design.

**Figure 2.** Information model for the software system implementation.

As for generalizing unifications (GU), it is necessary and important to have a wide range of creative perceptions to start their formation, with the possibility to return repeatedly to the Information Elements which are understood within the solution, as their proper utilization in a particular task is a condition for the quality of modules, algorithms, patterns, etc. that use them. The return from the Generalizing Unifications to original senses and semantics is extremely difficult in terms of clearly existing cognitive discomfort which is due to resistance to the destruction of a whole picture of the subject in the mind of the designer. The destruction of all integrity and completeness is always combined with a wrong choice, failure, or lost time and means. Therefore, such erosion should be excluded as much as possible.

The goal to create a concept of a task that forms an individual cognitive-contextual environment of the design to implement principal activities lies in creating possibilities to repeatedly and freely review information space of the design, which results in a clear picture of the task and the basis taking a definite shape for better creative solutions and evaluations. This approach is contrary to that when a solution is developed from certain options or, for example, finding the shortest path on the implementation graph. Moreover, regardless of whether or not the software graph is adjusted and weighted using expert assessments. Obviously, the proposed path is fundamentally different from the formal one where the focus is not on the essence of the problem, but on its artificial reducing to artificially unified forms. The main contributor to the cognitive-contextual design is reliance on the perception of the realized Information Elements and explicit data manipulation as the awareness is a
process of presenting internal data (sensory and motor categories) in an explicit form, which generates feelings, symbols, images and scenes [8].

Let us consider an example of the contextual-cognitive design of some attributes of a hierarchical file system, and take a similar example to compare, which illustrates the use of the COMPOSITE pattern [9]. It is best to begin designing with approximate assessments of the quality of the file system members without separating the functional side from the informational side. There are only five such members: File, Catalog, Get Data (GET), Put Data (PUT), Random Operation (Operation) (figure 3).

The main task of contextual-cognitive design is to preserve the cognitive integrity of the project, in other words, the perceptual coherence not only with formal, but also with real objects of the task. Therefore, we will place each of the design members on the Information Board [10] independently of each other (figure 3). Next, we will make meaningful links between the project elements (figure 3-b). It is clear that such links are only possible between some, but not all elements. The next step is to highlight emerging commonality among many project elements and finally, consciously generalize them in a formal manner, that is, to merge them into a group on a certain, definitely significant basis. We find that we can combine them into a composite Information Element - Component based on the overall coherence of the File and Catalog with the element Operation (figure 3-c).

Figure 3-d presents the structure of the COMPOSITE pattern which, as indicated in the [9], allows you to create the backbone of an application and express basic characteristics of the hierarchical file system using object-oriented concepts. When comparing the structure of the COMPOSITE pattern with the structure of the result from cognitive-contextual design, the commonality of the final solution is revealed, which evidences the validity of the cognitive-contextual approach. However, unlike the DP design in environment of the comfortable cognitive coherence, each Information Element is
perceived, individually recognizable and assimilated and therefore, reliable in decision-making and evaluation of results. At the same time, we note that the final structure of the hierarchical file system (figure 3-c) is not necessarily interpreted in an object oriented sense, but it only emphasizes the unifying right when generalizing on some grounds. In other words, the consciousness of the design in the environment that is based on the cognitive coherence will not be overloaded with various indirect and unutilized entities but will appear in a cognitively recognizable experimental environment where it is easier to accept a creatively specific and individually conscious decision.

3. Discussion
The advantages of this approach are obvious in terms of maintaining the cognitive coherence. Each element of the project is consciously embedded in its canvas or, on the contrary, is rejected while remaining available for further manipulations. The main focus here is that any simple or complex Information Element of the project passes through a kind of a perception filter and can, being recognized, be consciously used. The reverse situation of impossibility to be fully applied will occur when an Information Element is unknown or irresponsible in relation to the actual space of the task. For example, if the possibility to write in a file becomes known as a precedent, and a special relationship is called artifacts, it is not possible to make decisions in the abstract semantics until it becomes individually familiar in the context of a particular task. In other words, when the perception of the project Information Elements is difficult (due to formalization, unclear terminology, requirements to support established rules and formal actions), then the inevitable but expected result of the software design will be the loss of performance and quality of the Software System.

The usage of unified technologies such as PD requires the designer to:

- rely on existing architecture;
- form a project within an object oriented approach but not in an arbitrary way;
- have good skill in using such or equivalent pattern, i.e. learning. At the same time, until the chosen pattern becomes sufficiently learned or until the designer’s mind does not react negatively, the pattern will not be possible to apply in full.

So, in order to express the basic idea of cognitive-contextual software design as opposed to the Western structuralism tradition to rely on rules, stereotypes, patterns and generalizations, we should rephrase a well-known saying: it is necessary to invent your own "cognitive bike" every time so that you don't suddenly end up on someone else's crutches.

4. Conclusions
Presently, the software design can offer two main areas of practical implementation that are based on the strategy of contextual and cognitive design. Firstly, it is the design of hard-to-determinate systems that initially contain in the task data some factors of unclear complexity and ambiguity of decisions chosen. The main thing is that, when we adhere to the procedure for accumulating information elements of the arbitrary stage of the Software System lifecycle, we should not be afraid to make reckless assessments or actions but consistently expect clarification of the certainty of the decisions chosen while being aware that each such decision is only a temporary compulsory measure in the dynamics of the software design process. Secondly, it is the design of heuristic algorithms where the validity of the right decision is not critical in the choice of and search for operations, but for which the acceptability of the final results is only sufficient, focusing on the quality of which the algorithm itself is created.

References
[1] DeMarco T and Lister T 2017 Peopleware: productive projects and teams. Addison-Wesley.
[2] Paivio A 2017 A dual coding perspective on imagery and the brain. *Neuropsychology of visual perception* 203-16
[3] Köse D B, Morschheuser B and Hamari J 2019 Is it a tool or a toy? How user conceptions of a system’s purpose affect their experience and use International Journal of Information Management 49 461-74

[4] Yilmaz S, Daly S R, Seifert C M and Gonzalez R 2015 How do designers generate new ideas? Design heuristics across two disciplines Design Science 1

[5] Zhang X P, Nickels D, Poston R and Dhaliwal J 2018 One world, two realities: Perception differences between software developers and testers Journal of Computer Information Systems 58(4) 385-94

[6] Mendes F F, Mendes E and Salleh N 2019 The relationship between personality and decision-making: A Systematic literature review Information and Software Technology

[7] Balletta S and Domaneschi F 2019 Predication and cognitive context: Between minimalism and contextualism Ratio 32(3) 182-91

[8] Sergin V Y and Sergin A V 2019 Hierarchical model of perception without combinatorial explosion Journal of Higher Nervous Activity named after I.P. Pavlov 69(5) 629-54

[9] Johnson R and Vlissides J 1995 Design patterns. Elements of Reusable Object-Oriented Software (Addison-Wesley)

[10] Newell A and Simon H A 1989 Computer science as empirical inquiry: symbols and search Perspectives on the computer revolution 109-33