Life Cycle Concept Considered Harmful

Daniel D. McCracken(1) and Michael A. Jackson (2)

The authors were among the participants at an invitational conference held in September of 1980 at the Georgia State University to study the topic "Systems Analysis and Design: A Plan for the 80's."(3)

We found the conference and workshop stimulating and useful, and we support many of its processes and conclusions.

However, we came to the meeting unprepared for the stultifying effect of the organizers' decision to structure the group's work around the concept of a system development life cycle.

The life cycle concept as promulgated in preconference mailings was that systems development consists of the following ten steps:

1. Organizational analysis

2. Systems evaluation

3. Feasibility analysis

4. Project plan

5. Logical design (produces general design specifications)

---

(1) Daniel D. McCracken, Inc., New York

(2) Michael Jackson Systems Limited, London

(3) The published report, with the same title, is available from Elsevier North-Holland, New York, 1981, 553 pp. The meeting was organized by the Institute for Certification of Computer Professionals (ICCP), and sponsored by the Computer Society of the Institute of Electrical and Electronics Engineers (IEEE-CS), the Association for Computing Machinery (ACM), ICCP, and Georgia State University. The body of this paper is printed in the report under the title "A Minority Dissenting Position."

6. Physical design (produces detailed design specifications)

7. Program design

8. Implementation

9. Operation

10. Review and evaluation

Most attendees apparently found the life cycle concept comfortable. Several pointed out that there are many variations on the basic theme, but asserted that all life cycle concepts can be mapped onto each other. We would be interested in a demonstration of the latter assertion, which we doubt is true in any strict sense; if it is true, then we believe that the life cycle concept is probably, on balance, harmful. We adduce three groups of criticism.

1. Any form of life cycle is a project management structure imposed on system development. To contend that any life cycle scheme, even with variations, can be applied to all system development is either to fly in the face of reality or to assume a life cycle so rudimentary as to be vacuous.

The elaborate life cycle assumed as the basis for this conference may have seemed to be the only possible approach in the past when managing huge projects with inadequate development tools. (That it seemed to be the only choice, obviously did not prevent many such projects from failing.) It is foolish to use it as the basis for a certification effort that may be hoped to have a significant impact on the education of systems analysts for the next decade or more. During that time it seems highly probable to us that new development methodologies and implementation techniques will force--or rather permit--a complete re-evaluation of the assumptions built into the life cycle way of thinking.

2. The life cycle concept perpetuates our failure so far, as an industry, to build an effective bridge across the communication gap between end-user and systems analyst. In many ways it constrains future thinking to fit the mold created in response to failures of the past. It ignores such factors as the following, all of which are receiving rapidly increasing attention from both researchers and practitioners:

-- The widest possible development of systems by end-users themselves, limited only by the (rapidly improving) availability of development tools suited

to the purpose. This is, of course, the optimum response to the communication gap: eliminate the need for communication.

-- Heavy end-user involvement in all phases of the application development process--not just requirements specification, but design and implementation also.

-- An increasing awareness that systems requirements cannot ever be stated fully in advance, not even in principle, because the user doesn't know them in advance--not even in principle. To assert otherwise is to ignore the fact that the development process itself changes the user's perceptions of what is possible, increases his or her insights into the applications enviornonment, and indeed often changes that environment itself. We suggest an analogy with the Heisenberg Uncertainty Principle: any system development activity inevitably changes the environment out of which the need for the system arose. System development methodology must take into account that the user, and his or her needs and environment, change during the process.

What we understand to be the conventional life cycle approach might be compared with a supermarket at which the customer is forced to provide a complete order to a stock clerk at the door to the store, with no opportunity to roam the aisles--comparing prices, remembering items not on the shopping list, or getting a headache and deciding to go out for dinner. Such restricted shopping is certainly possible and sometimes desirable--it's called mail order--but why should anyone wish to impose that restricted structure on all shopping?

3. The life cycle concept rigidifies thinking, and thus serves as poorly as possible the demand that systems be responsive to change. We all know that systems and their requirements inevitably change over time. In the past, severely limited by inadequate design and implementation tools as we were, there was little choice but to freeze designs much earlier than desirable and deal with changes only reluctantly and in large packets. The progress of system development thus moved along a kind of sawtooth, with a widely separated set of points being designated "completion of implemention," "completion of first set of modifications," etc. The term "life cycle," if it has any linguistic integrity at all, seems an odd way to describe this process. To impose the concept on emerging methods in which much greater responsiveness to change is possible, seems to us to be sadly shortsighted.

Here, as sketches only, are two scenarios of system development processes that seem to us to be impossible to force into the life cycle concept without torturing either logic or language or both.

1. (Prototyping.) Some response to the user's earliest and most tentative statement of needs is provided for experimentation and possibly even productive use, extremely early in the development process--perhaps 1% of the way into the eventual total effort. Development proceeds step-by-step with the user, as insight into the user's own environment and needs is accumulated. A series of prototypes, or, what is perhaps the same thing, a series of modifications to the first prototype, evolves gradually into the final product. Formal specifications may never be written. Or, if specifications are needed, perhaps to permit a re-implementation to improve performance, the prototype itself furnishes the specifications.

This mode of operation is emphasized by some writers as being fundamental to the concept of Decision Support Systems; we believe it has even wider applicability.

2. A process of system development done by the end-user and an analyst in this sequence: implement, design, specify, redesign, re-implement. The user is provided with an implementation tool and one version of a system thought to be potentially useful. By a process of experimentation, in occasional consultation with the analyst, the user carries out--essentially in parallel--the following activities:

-- Learning the capabilities of the implementation tool.

-- Designing the desired system.

-- Specifying the desired system.

The analyst then works from the "specification," which in fact consists of a running functional model of the system, to produce a design that is implemented by a programmer in some conventional language.

* * * *

We claim that neither of these scenarios, nor many others that are readily imaginable, can be "mapped onto" the life cycle reproduced at the beginning of this note, or onto any other life cycle that is not vacuous. The life cycle concept is simply unsuited to the needs of the 1980's in developing systems.