Using Resource Constraints to Control the Incremental Development of Large Scale MIS Projects

SULEIMAN K. KASSICEH
and ROBERT S. TRIPP
The University of New Mexico

Incremental MIS design and development methodologies have received limited attention in the MIS literature. This paper describes incremental methodologies appropriate for very large scale MIS developments which can cost millions of dollars and take years to develop. This paper describes a “resource constrained” implementation planning process designed to control both the costs of incremental projects and the functional evolution of the system. The paper also discusses how the process was used in a very large scale system development.

Introduction

Development methodologies for MIS are constantly being enhanced and described in the literature. There is widespread recognition that a rigorous framework is necessary to insure that sound technical design principles are introduced in the MIS development process. To meet this need, the traditional systems development life-cycle (SDLC) method has evolved. The SDLC calls for the development of a system by following a series of rigorous steps in sequence. The shortcomings of following this sequential process have been documented by Appleton (1986), Synder and Cox (1985), Ahituv and Neumann (1984), Juergens (1977), and King (1982), among others. These authors indicate that it is not practical to attempt to establish detailed requirements specifications and detailed system designs before beginning the development of any portion of the system. Tripp and Filteau (1987) discuss how these phases can take years to finish on large scale systems, and that before they can be completed, environmental factors change, requiring a change in the requirements and design of the system. As a way to avoid some of these problems of the SDLC, prototyping has been offered as at least a partial solution (Bally, Brittan, & Wagner, 1977);

In building very large scale MIS which span years of development as well as cost in the millions of dollars, neither of the above approaches is likely to produce desired results. The "build-it-twice" prototype approach is too expensive for these large projects and the level-by-level SDLC approach requires too much time to complete all steps before software products are released to users. As a result, several authors have suggested refinements to the SDLC model to facilitate the development and release of increments of the total system.

Boehm (1981) indicates that the main advantages of the incremental development approach over the prototype and SDLC approaches are that the increments are easier to test than the intermediate products associated with the SDLC and that the use of increments by users provides a less expensive way of incorporating user experience than the "build-it-twice" prototype approach. Using the incremental development approach requires the decomposition of the super-system into subsystems which are of manageable size and can be developed somewhat independently. As those subsystems are developed, the integration of the components into a unified entity must take place.

The remainder of this paper describes an incremental development implementation planning process which was used to: (1) decompose a very large MIS - the Air Force Requirements Data Bank - into subsystems, (2) prioritize those subsystems for development, and (3) control the costs and direction of the multi-year development.

Incremental Design Methodology

In multi-million dollar, multi-year MIS developments, it is absolutely necessary to develop and release products to users on an evolutionary and frequent schedule due to the length of the development effort which subsumes that people, conditions and systems that existed at the start of the development are not the same as when it is in the final stages. Moreover, the development team must reinforce users’ perceptions that the development goal is being met. Users cannot and should not be expected to wait for lengthy periods for improved capability. Equally important, users and top corporate management need constant reassurance that investments of great amounts of resources are, in fact, paying off.

The following steps in the iterative development process are recommended:

**Design Step 1**: A “fairly” detailed top-down analysis of the functional requirements of the entire super-system is conducted. The intent of this analysis is to describe the requirements of the entire system in some detail. The goal of this phase is to obtain knowledge of the objectives and boundaries of the system and to determine the feasibility of building the super-system.

**Design Step 2**: The super-system is broken down into smaller subsystems for fairly rapid development and release to users. As Boehm (1981) indicates, each of these increments or subsystems should be developed following the standard steps outlined by the SDLC. Juergens (1977) maintains that the manner in which the development is partitioned is critical to the success of the system. He indicates that large systems can be decoupled vertically and horizontally to form subsystems for development. Horizontal decoupling separates the development into relatively independent subsystems which do not have substantial interaction. These subsystems can then be developed while planning for the eventual interfaces takes place. This process needs to be coupled with an evolutionary database design which insures that the needs of evolving software applications are met (Tripp & Filteau, 1987). Vertical decoupling separates portions of the development within a functional area and allows a portion of the function to be developed and used while the expansion and enhancement of the module is scheduled for later development. This decoupling reduces risks and allows for incremental implemen-
Related Content

Military Applications of Natural Language Processing and Software
www.igi-global.com/chapter/military-applications-natural-language-processing/44495?camid=4v1a

Influence of Personality on Programming Styles an Empirical Study
www.igi-global.com/article/influence-of-personality-on-programming-styles-an-empirical-study/145393?camid=4v1a

Delegation
www.igi-global.com/chapter/delegation/29001?camid=4v1a

The Expert's Opinion
www.igi-global.com/article/expert-opinion/50947?camid=4v1a