The development of information systems (IS) emerges from the need for more expressive designs that could work well in diverse situations. Over the years, several methodologies have been proposed that offer tangible alternatives that capture solutions to the problems in context, and to a limited degree for those that could potentially occur in time. This paper describes the advantages and pitfalls within each of the known approaches and the reasons why no single methodology has gained general acceptance. Efforts to integrate these approaches often require one to determine the right mix of features (Lyytinen, 1987). A case study of a large manufacturing organization from the midwest is presented and a new methodology that attempts to blend in pertinent features of the existing models of IS design is proposed.

A survey of the literature indicates that information systems (IS) are subject to several deficiencies including lack of user involvement in the process; failure to incorporate cognitive, behavioral, and organizational issues into the design; poor user interfaces; etc... (Baroudi, Olson, & Ives, 1986; DeBrabander & Edstrom, 1977; Doll & Torkzadeh, 1990; Edstrom, 1977; Ein-Dor & Segev, 1978; Franz & Robey, 1986; Larcker & Lessig, 1980; Robey & Farrow, 1982; Robey, 1983; Zmud, 1979). In addition, Lyytinen (1988) points out that as many as one-half of the information systems developed are subject to severe problems that could be considered as failures, and that such failures may occur in the design/development process as well as in use and operations.

Following the framework by Ives, Hamilton, and Davis (1980), Lyytinen (1987) surveyed and categorized problems of the IS process. For example, six major problems in the IS development process are (1) ambiguous, narrow, and conflicting goals; (2) technology that restricts choices and is overly susceptible to change; (3) problems in economy due to inaccurate calculations and weak foundations; (4) process features encompassing poor communication, quality control, and domination by the systems analyst; (5) neglect of behavioral and organizational issues; and (6) a highly rationalistic self image (See also Anderson, 1989; Cheney & Dickson, 1982; DeBrabander & Thiers, 1984; Gorla, 1989; Huber, 1983;
Jenkins, Naumann, & Wetherbe, 1984; Mahmood, 1990; Zviran, 1990). Of the six problems listed, further study by Lyytinen (1988) revealed that the problems pertaining to goals, process features, and organizational issues are the most common.

Also included in Lyytinen (1987) are five use and operations process related problems: (1) operations problems such as awkward interfaces, slow and unreliable design, and difficulty in usage; (2) data problems such as incorrect, irrelevant, incomprehensible, and missing data; (3) conceptual problems such as ambiguity and/or misunderstanding; (4) people related problems such as power shifts and negative impact on work; and (5) complexity in understanding, maintenance, and use (See also Alter, 1980; Cerullo, 1980; Shneiderman, 1981). Lyytinen (1988) adds that conceptual and data problems occur frequently.

There are several methodologies that facilitate the IS process and help alleviate its deficiencies. Among the more widely used are the Systems Development Life Cycle (SDLC) and Prototyping. Others include the Pragmatic Input/Output Constructive and Operational model (PIOCO) (Iivari and Koskela, 1987), the Evolutionary Design model (EDM) (Lucas, 1978), the Organizational Change model (OCM) (Alter & Ginzberg, 1978; Alter, 1980), the Bargaining model (Kubicek, 1983), and the Discourse model (Checkland, 1981; Lanzara, 1983; Lanzara & Mathiassen, 1985). McFarlan and McKenney (1983) observe that a given methodology may be more appropriate for a particular situation than another, and that the type of methodology chosen may impact the success of an information system. However, in practice, methodologies selected for use are often inconsistent with the given situation (Saarinen, 1990). Efforts to integrate the known methods require the determination of the right mix of features ideal for the given situation (Lyytinen, 1987). However, in reality methodologies are rarely integrated effectively (Saarinen, 1990).

The purpose of this paper is two-fold. First, we examine each methodology to determine its key advantages and pitfalls. Such a description is essential to assess whether the selected methodology addresses or is capable of addressing the aforementioned design/development and use/operations problems. Second, we attempt to integrate the existing methods by proposing a new methodology, and we examine the outcome of our proposed method by presenting a case study.

This paper is organized into five sections. The first section provides the conceptual background wherein critiques of each methodology are detailed. An outline of the existing system at the manufacturing company in this case study is provided in the second section. The third section deals with the proposed methodology and a discussion of its features. Section four provides the lessons learned, and the conclusions are provided in section five.

**Conceptual Background**

Classification of methodologies is a fascinating subject of study in its own right. A good classification provides better insight about the similarities and dissimilarities between methodologies. We adhere to the classification used by Lyytinen (1987) because of its clear approach. Thus, methodologies are classified as engineering, learning, or dialogue models. Engineering models are those that approach IS development from the perspective of the technical aspects of the system: these models include SDLC, prototyping, and PIOCO. Learning models are those that approach IS development as an individual and group learning process. Included in this category are evolutionary design and organizational change models. Dialogue models are those that approach IS development from the perspective of bargaining and inquiry. The bargaining and discourse models fall in this category. In the following section, we present critiques of several existing methodologies with supportive literature. For additional information, refer to Lyytinen (1987, 1988).

**Engineering Models**

**System Development Life Cycle (SDLC).** The SDLC is functionally designed assuming rational individual behavior and represents the system through a series of stages of refinement and transformation. These stages may be classified in several ways (see, for example, Awad, 1988; Hussain & Hussain, 1985; Lucas, 1986; Martin, 1991; Taggart, 1990; Zmud, 1983). Schemer (1987) groups the stages into four major classes that include (1) specification of problems, system and software requirements, and the conceptual design; (2) development of detailed design, coding, testing, and establishment of operational procedures; (3) implementation of acceptance tests, user training, and conversion; and (4) operation and maintenance.

Requirement specification is the most crucial phase in the life-cycle approach. Methodologies and techniques that provide requirement specifications include: (1) Structured Systems Analysis (SSA) (DeMarco, 1978; Gane & Sarson, 1979; Mendes, 1980);