Chapter 12
Back to Basics:
In Support of Agile Development

Roy Morien
Naresuan University Language Centre, Thailand

ABSTRACT

Massive failures of software development projects have been recorded in the literature, and particularly in the popular press, over the years. Yet, rarely if ever have we seen any objective, detailed analysis of the causes of these failures. Indeed, we usually can only surmise how the projects were managed or what the development methodology or approach was. This chapter analyses some aspects of software development projects and development methodologies in terms of the success or failure potential of these methodologies. The conclusion arrived at is that the system development methodologies handed down since the late 1970s as the preferred development approach, generally known as Structured Methodologies, based on the Structured Design Life Cycle methodology (SDLC), bear the seeds of their own failure. It is asserted that they cannot succeed because of the inherent nature and assumptions embedded in those methodologies. After some analysis of these assumptions, considered to be highly flawed and unworkable, the now not so recently published Agile Development methodologies are discussed and proffered as a workable and inherently successful approach to software system development.

INTRODUCTION

The message in this chapter is simple, but presumably controversial to the extreme that it is almost ideological in the on-going 'methodologies' battle in the IT industry. This battle is between what may be termed the 'Heavyweight'' methodologies, and the 'Lightweight' methodologies used in software system development and project management. Sometimes the 'heavyweight' methodologies are termed 'industrial strength methodologies', emphasising their apparent strength in process control, especially in large systems development. The implication inherent in this terminology is that the 'lightweight' methodologies are weak and inappropriate to 'real world' large projects.

Notwithstanding this terminology, the message in this chapter is that the traditional approaches to software systems development, first published under the heading of Structured Design Lifecycle Approaches (SDLC) in the late 1970's (deMarco, 1979), (Yourdon & Constantine, 1979), (Gane &

DOI: 10.4018/978-1-4666-6026-7.ch012

Sarson, 1977) have over time proven to be at best flawed and at worst disastrous in ensuring successful system development outcomes. Yes, many systems have been developed that do work, but the contention is that even these 'successful' systems would have had more successful outcomes; lower cost, shorter development times, greater business value and better user acceptance if a 'lightweight' development approach had been applied.

That 'lightweight' approach is now generally known as Agile Development, and comes in various guises, such as Extreme Programming (Beck, 1999), Scrum (Schwaber & Beedle, 2001), Dynamic Systems Development Methodology (DSDM) (DSDM Consortium, 2012) to name the most prominent, and Evo (Gilb,1976), to name one of the oldest. These methodologies all have their roots in development approaches such as Software Prototyping (Naumann & Jenkins, 1982), Rapid Development (RAD) (McConnell, 1996) and Rapid Application Development (Martin,1991) of much earlier years.

ORGANIZATION OF THE CHAPTER

The chapter develops the theme that to gain support for the adoption of Agile methods in software system development and project management, it is necessary to identify and explain the weaknesses in the traditional development approaches to be able to counter the usual objections to Agile Methods. This is done in the sections discussing, and refuting, the idea of 'Big Systems' and the preferred development methods for such systems. The weaknesses and failure of the underlying assumptions supporting the traditional Waterfall Approach are then discussed.

This refutation of traditional development practice finishes with a discussion of the justifying Fear of Change that is still felt, even in the face of significant changes to all or most other development dimensions and the development of a significant marketplace in development tools, application generators and the like.

The chapter continues into a description of what is seen as the true nature of software projects, as being 'people-centric', and 'learning organisations'. The thesis is that these types of group activities need to be managed differently to projects in civil engineering and construction, which seems to be the preferred paradigm for a 'project management', regardless of project type. More appropriate means of gathering and specifying requirements are addressed, as a lead in to a discussion on iterative and incremental methods, and the perceived benefits inherent in such approaches.

Finally, the chapter introduces a number of what may be called 'reference disciplines' in an attempt to provide an intellectual and theoretical underpinning to a development approach that has been largely promulgated based on practitioner experience, rather than having a principled and theoretical basis.

A contentious place to start this discussion is found in a recent government inquiry in the state of Queensland in Australia which investigated the substantial failure of the development of a payroll system for the Health Department in that state. In summary, the $5,000,000 inquiry looked at a project that initially was quoted as a development cost of about $6,200,000 and has apparently exceeded $2 billion in sunk cost and remediation costs, and operating costs blown out due to poor design causing substantial operating costs. This figure of $2 billion seems to have been modified to $1.2 billion in later announcements by the Queensland Premier, perhaps indicating the inability to estimate the actual cost. Substantial business disruption occurred due to the failure of the system to pay wages and salaries correctly to numerous employees that lead to street demonstrations by affected staff protesting the situation; such was the seriousness of the problem.