Chapter 20

Prioritizing Packaged Software Implementation Projects:

The Significance of Gaps

Nicholas J. Rowland
Pennsylvania State University, USA

ABSTRACT

This chapter examines the dynamics of prioritizing implementation projects. Building on the notion of "fit-gap" work, this chapter emphasizes the significance of "de-prioritization" as a practical technique for managing Enterprise Resource Planning (ERP) implementation projects. "Fit-gap" is a term that resonates with current academic and professional discussions concerning the use of customization and work-arounds necessary to coax suboptimal implementations into functioning properly as the systems age. These are not idle matters given the near irreversibility of ERP projects once initiated and the reported high probability of failure following implementation. Drawn from in-depth interviews and internal documents collected from a multiyear organizational case study of ERP in an institution of higher education, this chapter reports on various uses, interpretations, and consequences of prioritization techniques used to manage implementation projects. In practice, the idea that complex software implementations can be theoretically reduced to mere gaps in fit serves to obscure the political conflict and ambiguous economic accounting that underlie committee work devoted to identifying gaps, deliberating on possible fits, and then prioritizing which gaps are fit immediately and others scheduled for fit later on. In conclusion, while fit-gap committee work is openly intended to result in fewer customizations overall, de-prioritization, as a management technique, appears to "remove without removing" agenda items from the implementation schedule. The upshot for managers: placing such decisions in purgatory delays indefinitely investments of time and finances into customizing new software to fit old policies, and all the work-arounds necessary to shore-up any lingering idiosyncrasies.

DOI: 10.4018/978-1-4666-4153-2.ch020

INTRODUCTION

The promise of packaged software solutions such as Enterprise Resource Planning (ERP) very much hinges on determining how to make them work on-the-ground (Wagner & Newell, 2006). The product's vendors guarantee flexible software packages (Pollock & Williams, 2008), often citing the "80/20" rule, which implies that ERP systems are roughly 80% delivered with the remaining 20% left to the discretion of local programmers (Pollock, 2005, p. 7). However, as the literature documents, malleability brings with it the burden of "fitting" work (Gasser, 1986), "work-arounds" (Pollock, 2005), and "technological adjustments" (Pfaffenberger, 1992). Therefore, it appears that implementing packaged software systems implies managing the interpersonal, financial, and temporal aspects of initial installation, ongoing maintenance, and the nearly inevitable idiosyncrasies inherited from previous system development efforts (Light, 2005; Thomas, 1994).

Building on Gasser's (1986) notion of "fitting" work and Rowland's (2008) emic term "fit-gap," this chapter examines the process and consequence of prioritizing implementation projects with special emphasis on "de-prioritization" as a practical technique for managing installation. As it happens, organizations follow elaborate formal and informal procedures in their efforts to shape the progressive implementation of ERP systems (Pollock, 2005; Wagner & Newell, 2006). The reason is simple: packaged software do not fit organizations that wish to install them (Pollock & Cornford, 2004). Therefore, a cocktail of formalized fit-gap procedures and on-the-fly work-arounds is the hallmark of packaged software implementation projects.

To manage the numerous misalignments between the organization's extant structure and packaged software off-the-shelf, which are bound to emerge during the implementation process, project managers assign key functional and technical employees to sit side-by-side on "fit-gap committees." Borrowing terminology from those charged with implementing and maintain such systems in higher education, Rowland and colleagues (Rowland, 2008; Gieryn, 2008) describe fit-gap work as a formalized process devoted to

- Identifying gaps,
- Deliberating on possible fits, and then
- Prioritizing which gaps are fit straightaway while others get de-prioritized and remain agape.

Conceptually, the fit-gap idea is a valuable tool during implementation. However, the idea that complex software implementations can be theoretically reduced to mere gaps in fit serves to obscure the political conflict and ambiguous economic accounting, which are also significant elements of fit-gap work—especially in relation to de-prioritizing some gaps for fit (presumably) later on (on politics and power during implementation see Markus [1983] and Robey and Markus [1984]).

The work-around becomes a resource to shore-up lingering misfits amid the minutiae of the new machine (Gasser, 1986; Pollock, 2005). But not all gaps in fit are possible to identify despite the best efforts of fit-gap committees. Presumably, this is because committees are forced by necessity to adopt an aggregate view of the organization. This perspective brings with it a modicum of abstraction to the otherwise infinite nuance of everyday life in the organization, which is not unlike the view described by Scott (1999) in *Seeing like a State*. Returning to work-arounds, Pollock (2005) shows readers how local technicians overcome system misalignment by techno-scientific patchwork designed to mend the interface between the new system and extant practices and already established support software. Also referred to as kludges, work-arounds make the systems "work" for the moment, at least until the next round of vendor updates arrives. Pollock (2005) views work-arounds as emblematic of packaged software

Related Content