Enterprise Resource Planning (ERP) Maintenance Metrics for Management

Celeste See-pui Ng
Yuan-Ze University, R.O.C.

INTRODUCTION

A typical packaged software lifecycle, from the client-organization perspective, is packaged software selection followed by implementation, installation, training, and maintenance (that includes upgrades). Traditional software maintenance has been acknowledged by many researchers as the longest and most costly phase in the software lifecycle. This fact is no exception in the ERP packaged software maintenance context (Moore, 2005; Whiting, 2006).

According to Ng, Gable, & Chan (2002, pg. 100) ERP maintenance is defined as "post-implementation activities related to the packaged application software undertaken by the client-organization from the time the system goes live (i.e., successfully implemented and transported to the production environment), until it is retired from an organization's production system, to keep the system running; adapt to a changed environment in order to operate well; provide helps to the system users in using the system; realize benefits from the system (best business processes, enhanced system integration, cost reduction); and keep the system a supported-version and meet the vendor's requirements for standard code. These activities include: implementing internal change-requests (initiated by an ERP-using organization's system users and IT-staff); responding or handling user-support requests (initiated by an ERP-using organization's system users); upgrading to new versions/releases (introduced by the vendor); and performing patches (support provided by the vendor)."

In order to achieve the abovementioned maintenance objectives of keeping the ERP system running, adapting the system to a new operating environment, and ensuring the system up to the vendor's requirement for standard code; and realizing benefits such as competitive advantages from the system, the IT department staff has to collect some metrics or relevant data on patches and modifications done to the ERP system so that they can know or can tell the status and the performance of their maintenance activities. The authors in Fenton (1991), Fenton & Pfleeger (1997), and Florac (1992), agree that software maintenance data are useful for planning, assessment, tracking, and predictions on software maintenance. Although, there is a lot of literature on ERP, we find almost no literature on ERP maintenance metrics. Thus, this text is meant to provide some fundamental metrics on ERP patches and modifications which could be useful for ERP maintenance management in order to answer questions on the state of their ERP system, their patch implementation costs, and the ongoing maintenance costs for their previous modification or custom development.

BACKGROUND: METRIC

The *IEEE Standard Glossary of Software Engineering Terminology* (1990) defines a metric as a "quantitative measure of the degree to which a system, component, or process possesses a given attribute." Based on the definition, this text interprets a metric as being derived from data, and as quantifiable, meaningful, and used for strategic, tactical and/or operational purposes. Data (or data item), in turn, is defined as a quantitative indication of the extent, amount, dimension, capacity, size, or characteristic of particular attributes of a task or activity in a process. It can be collected using forms (e.g., change request form, change report, software engineering report), interviews (with the users, testers, programmers, analysts, managers), and via computerized systems (e.g., the in-built change management system in ERP, change request database). A goal/question/metric (GQM) paradigm is a systematic way of collecting predefined data, with intended goal(s), and the associated sets of predetermined questions, in order to derive the anticipated measurable metrics. Basili and Weiss (1984) advocate this methodology for collecting valid data. In GQM, which is also known as the top-down approach, the timing (in terms of the software life cycle or activity), interviewees, and reasons for collection are all predetermined.

The literature reports that successful use of measurement/metric program avoids recurring errors (Ebert, Dumke, Bundschuh, & Schmietendorf, 2005), improves software maintenance processes at Burrough Corporation (Rombach & Ulery, 1989), and Motorola (Smith, 1993), and improves product quality at AT&T (Fenton & Pfleeger, 1997).

On the flip side, a known metric – together with the context for its interpretation – can determine what data might be collected (Rombach & Ulery, 1989). There are three main purposes of metrics: assessment, prediction, and control.

Table 1. Main characteristics and purpose of three main metric categories

Metric category	Main characteristic	Purpose	Example of ERP maintenance metric
Assessment	Informative (about person, process, object); may be used in decision making or for controlling purposes	Retain knowledge	Programmer ID, problem description, description of changes, issues of consideration, patch ID
Prediction or decision-making	Conclusive; most likely derived from the assessment and control metrics; usually describes what should be done	Make decision, planning and estimation	Estimated maintenance time, quotation for a maintenance request, action to be taken, maintenance request type, projected availability
Control	Indicative (indicating that something needs to be done); most likely used to pinpoint that a particular decision needs to be made; usually requires data to be collected over a period of time; usually has some attached baseline value	Monitor performance, track progress, identify problem	Problem status, approved by, accepted by, maintenance request ID, time of problem occurrence, resolution impact

Table 2. Application of software metrics in practice

Purpose	Case name	Metrics used	Goal
Assessment	Hewlett-Packard (Wood, 2003).	Event chronology, problem symptoms, diagnostic information, release version information	Problem analysis and resolution
Decision-making	NASA's Mission Operations Directorate (Stark, Durst, & Vowell, 1994)	Previous project delivery rate	Estimate a test schedule
	Hewlett-Packard to (Grady, 1994).	Defect trend	Determine time to release a product
Control and monitoring	NASA's Mission Operations Directorate (Stark, et al., 1994)	Earned value management technique: project cost and schedule	Monitor project cost and schedule performance
		Defect density	Track quality in subsystem, efficiency in testing, and backlog of fault
	Hewlett-Packard (Grady, 1994)	Code size and time	Monitor project progress
	Bull's Arizona (Weller, 1994)	Effort, resources, product size, estimated completion date and defect detected	Manage project and improve project planning
	Siemens (Paulish & Carleton, 1994)	Defect rate (i.e. number of defect/ product size)	Measure performance
		Product size and effort (i.e. product size/effort)	Project productivity
		Productivity gain, error detection rate, and reduction in time to market	Measure software process improvement