Balancing uncertainty of context in ERP project estimation: an approach and a case study

Maya Daneva* †

Computer Science Department, University of Twente, Drienerlolaan 5, 7500 Enschede, The Netherlands

SUMMARY

The increasing demand for Enterprise Resource Planning (ERP) solutions as well as the high rates of troubled ERP implementations and outright cancellations calls for developing effort estimation practices to systematically deal with uncertainties in ERP projects. This paper describes an approach—and a case study—to balancing uncertainties of context in the very early project stages, when an ERP adopter initiates a request-for-proposal process and when alternative bids are to be compared for the purpose of choosing an implementation partner. The proposed empirical approach leverages the complementary application of three techniques, an algorithmic estimation model, Monte Carlo simulation, and portfolio management. Our case study findings show how the ability of our approach to model uncertainty allows practitioners to address the challenging question of how to adjust project context factors so that chances of project success are increased. We also include a discussion on the implications of our approach for practice as well as on the possible validity threats and what the practitioner could do to counteract them. Copyright © 2010 John Wiley & Sons, Ltd.

Received 11 February 2010; Accepted 11 February 2010

KEY WORDS: Enterprise Resource Planning implementation; portfolio management; project effort estimation; COCOMO; Monte Carlo simulation

1. INTRODUCTION

Enterprise Resource Planning (ERP) systems are packaged software solutions, the key function of which is to support intra- and inter-organizational coordination processes in business units within a business. They are the vehicles modern organizations use to achieve true business connectivity, a state in which everyone knows what everyone else is doing in the business all over the world at the same time. Analyst firms, such as AMR Research [1], Forrester [2], Gartner [3], indicate that the demand for inter-organizational ERP solutions and the investments in ERP assets are continuously growing which, in turn, suggests that ERP represents one of the significant software (and software consulting) markets. We observe that ERP implementation has spawned an independent industry with firms providing consulting services exclusively and generating total revenue of billions of dollars. However, because of their relatively short history of existence, the business practices of requesting proposals for ERP implementation services, of bid preparation, and of pricing are not well understood. ERP adopters find themselves, increasingly more often than ever, caught in a situation in which they need to get initial estimates quickly, mostly based on uncertain assumptions,
to compare bidding information from competing implementation service providers, and to initiate negotiations with them. For ERP adopters to be adequately prepared for this endeavor, they need to acquire knowledge (i) on how each consulting firm arrived at the bidding price and (ii) on how realistic the effort estimation figures, which one sees on the bidding document, are. However, at that early stage of requirements engineering (RE), more often than not, uncertainties of context interfere greatly with adopter’s ability to assess to what extent the price they receive from the bidding document matches their organizational realities. Today, it is well known that a typical ERP project includes diverse configurations, each of which matches the needs of a unique stakeholder group, which, in turn, implies the presence of cost drivers unique to each configuration. Moreover, at the time of bid preparation (that is, the stage of very early requirements), consulting firms would have a relatively low level of awareness of (i) what new project activities (e.g., identifying and analyzing capability gaps, investigation and mapping of configuration options) are to be added in order to plan and manage the ERP project in a particular adopting organization, and (ii) what the ERP adopter’s context factors are that drive effort for these new activities. Yet, ERP adopters must act quickly and find a way to balance uncertainties of their contexts at that very early stage. Often, what both consultants and adopters do in the face of this situation is to make plans on average assumptions. This, however, may only aggravate the problem, as Savage says in [5] ‘average assumptions will be wrong on average’. In ERP settings, being ‘wrong on average’, may well mean a negative impact on the business. Examples of underestimated ERP projects [6–8] suggest that making average assumptions might have a profound long-term effect on the ERP adopter, meaning a failure to meet growth targets [7], a loss of market edge [6], or even a bankruptcy [8].

In this paper, we make an attempt to alleviate the ERP adopter’s need for balancing uncertainties in ERP project estimation as part of the ERP RE process. We present a case study of how trade-offs among contexts uncertainties could be achieved from the perspective of the ERP-adopting organization. Our objective is to provide ERP adopters with a support vehicle to help them (i) get an increased understanding of the impact of their specific context on project scope, effort, and delivery dates, (ii) reason about the resources to be consumed in a project, and (iii) become aware of how uncertainties of project context matter in terms of project success. We design a multi-technique-based approach that rests on decomposition of a large ERP project into subprojects and algorithmic estimations on them, where the role of uncertainty of context characteristics is analyzed by using probability distributions. Our case study research demonstrated that when incorporating traditional effort/duration modeling concepts, portfolio management concepts, and variation concepts, it is possible to find a tradeoff between ERP projects requiring more effort than expected and those requiring less. We identified project context characteristics, which, when changed, can add up to a successful ERP implementation. The major practical implication of our results is that an ERP adopter (i) no longer has to live with consultants-provided estimates and (ii) can make informed decisions such as how to steer the project is a way that can cause a desired effect on important project outcomes.

The remainder of this paper proceeds as follows. In Section 2 we provide background information on the uncertainty challenges in the ERP effort estimation practice, the solutions published in the literature in information systems management and software economics and measurement, and the difficulties in using these solutions. In Section 3, we present our approach and motivate our design choices in each step of its development. Section 4 describes the case study research method we chose. It provides insights into the choices we made along the way and into the empirical data for our analysis. Section 5 presents a discussion on our results and their implications for research and practice. Section 6 deals with validity concerns and Section 7 presents concluding remarks.

2. BACKGROUND AND RELATED WORK

2.1. Specific sources of uncertainties in ERP project estimation

This section discusses uncertainty of ERP projects from effort estimation perspective that helps the readers to understand the remainder of the paper and to avoid misunderstandings. Our belief...
is that for ERP adopters to successfully deploy an effort estimation technique, they must first understand, from the standpoint of software effort estimation, (i) how ERP projects are different from custom software development or from other types of complex information systems (e.g., workflow management systems), and (ii) what sources of uncertainty are specific to ERP project context. We researched these two questions by carrying out a structured review [9] of the existing literature. We used the systematic process described in [9] to select the related literature sources presented in this section. The performed review consisted of a broad search of academic and practitioners’ information sources. We carried it out by using several electronic indexing services (namely, ACM Digital Library, Google Scholar, Citeseer library, and IEEEExplore) on 16 December 2008. A set of key words was used: Enterprise Resource Planning (ERP), budget estimation, effort estimation, project estimation, cost estimation, duration, size estimating. We also used some alternative terms for Enterprise Resource Planning: ERP, COTS, enterprise systems, Supply Chain Management systems, Customer Relationship Management systems, Interenterprise systems. We traced the references in the identified papers to get access to other relevant sources. We reviewed the abstracts and the conclusions of the identified documents in order to determine their relevance to our research. We deemed 17 papers candidates for inclusion in this section. We make a note that our search yielded more papers discussing the various aspects of ERP costing, however, only 17 papers (i) compared ERP systems with other project types and (ii) explicitly treated the topic of uncertainty in ERP effort estimation. These two criteria were decisive for the inclusion of the papers in this section. Below, we aggregate findings from the 17 publications.

Our observation from the review of these sources [6, 10–25] is that ERP effort estimation practitioners and researchers discuss the project uncertainties challenges from at least six perspectives, each of which entails specific issues:

- **ERP solution ownership:** This perspective is concerned with the questions of who shares and owns the ERP solution and who has the authority to make decisions over (which parts of) the solution. ERP projects do not necessarily have an identifiable owner at overall-solution-level because the solution is shared among business units or sometimes independent companies [6, 16, 21, 23]. Parts of the system are owned by different units each bearing part of the costs and each absorbing part of the benefits, whereby the cost and the benefits are not evenly distributed among the business units. This poses a unique challenge to effort analysts at bidding stage, because the partner companies rarely agree that early in the project on each company’s exact amount of participation in the cross-organizational ERP solution implementation.

- **Complexity:** This perspective addresses the ways in which the ERP solution is complex. Evidence from case studies in the literature [6, 10, 16, 23, 25] suggests that the growing complexity of the inter-organizational business processes means growing complexity in the ERP solution that embeds these processes. At early requirements stage, the client organization has little or no knowledge of the relative complexity of the parts of the shared ERP solution and the specific ways in which one part might be more complex than another. This can lead to gross inaccuracies when sizing the ERP solution to be implemented.

- **Architecture boundary:** This perspective is about how much architecture is in the project scope [6, 21–23]. ERP systems are not ‘built’ in the sense that a master architect envisions the parts and their relationships; rather they evolve into existence and change through their life cycles as new shared pieces of functionality are built, existing intra-organizational systems connect to become shared, and shared parts of the system are disintegrated as soon as needs of sharing business processes and data disappear.

- **Configuration:** This perspective is concerned with ERP configuration-specific challenges [6, 13–15, 17, 18, 22, 23]. Configurations create system capabilities far beyond the sum of the ERP components’ individual capabilities, which allows the resulting system to qualitatively acquire new properties as a result of its configurations. Each one implies the presence of cost drivers unique to each configuration.

- **Project duration:** This perspective addresses the concept of project duration being assumed by the partner companies client organization [11, 19]. An ERP project delivers a system that
Table I. Mapping of papers onto the perspectives from which uncertainty is viewed.

<table>
<thead>
<tr>
<th>Paper Reference</th>
<th>ERP solution ownership</th>
<th>Complexity</th>
<th>Architecture</th>
<th>Configuration</th>
<th>Duration</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>[6]</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>[10]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[12]</td>
<td>x</td>
<td>y</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[13]</td>
<td>x</td>
<td>y</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>[14]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[15]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[16]</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[18]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[19]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[20]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[21]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[22]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[23]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[24]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[25]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

is incomplete once the ERP project is over, because an ERP solution must mirror rapidly changing business requirements, and hence be adjusted regularly to accommodate current business needs.

- **Size**: This perspective is concerned with the concept of size being assumed [6, 12, 17–19, 21–25]. For example, Janssens et al. [12] present a concept of size that is related to the activities in an ERP project. More examples of notions of ERP size can be found in an earlier publication by the author [6]: ‘size’ can be referred to as an attribute of the implementation tasks (e.g., ‘size’ is defined as the number of ERP transactions to be configured), as an attribute of the ERP user base (e.g., the number of users), or as an attribute of the ERP functionality (e.g., number of function points (FP)).

In Table I we present whether each of the 17 publications discusses or only gives hints to issues pertinent to each of the six perspectives. The x-sign in the table means that a paper provides explicit statements about issues related to the perspective and the y-sign means that the authors of the paper give hints only. (An empty cell is to indicate that a paper does not discuss a certain perspective.) In Section 2.2, we will see that the existing techniques for handling uncertainty in effort estimation only partly address the challenges related to the six perspectives.

### 2.2. Related work: current approaches to uncertainty of software project context

The issue of how to treat uncertainties has been approached within a variety of areas, including nuclear plants, space shuttle vehicles, financial investment, and security. In our research, we chose to focus on project-uncertainty-handling approaches within (i) ERP implementation and (ii) software economics. Similar to the approach we used in Section 2.1, we applied the systematic process [9] to search and select the related work presented in this section. We carried out a broad search of academic and practitioners’ information sources using the four bibliographic databases we used in Section 2.1 (ACM Digital Library, Google Scholar, Citeseer, and IEEExplore). The key words we used were: ERP implementation, software economics, software uncertainty, project uncertainty, context uncertainty, uncertainty approach, uncertain effort estimation, uncertain cost drivers. As in the earlier search (Section 2.1.), we traced the references in the identified papers to get access to other relevant sources. To determine the relevance of these sources to our research, for each one, we reviewed the abstracts and the conclusions.

To build on the knowledge of the ERP implementation community (represented mainly by management science scholars) and the software economics community, we studied the ways on
which uncertainty-handling solution approaches are designed. Below we provide a summarized review of these solutions along with a discussion on why they bring only a partial answer to ERP adopters in the bidding stage. We make a note that the list of solutions mentioned in this section does not pretend to be complete. It should be considered as a list of illustrative examples only of what could be found in the software engineering and management science literature. We explicitly state that the research task of carrying out a full systematic review is a major undertaking, which is out of the scope of present paper.

2.2.1. Related work from the ERP implementation literature. Quantitative studies by management science scholars, for example Amram and Kulatilaka [26], Taudes et al. [27], Ranganathan and Brown [28], Bardhan et al. [29], De Reyk et al. [30], Verhoef [31], Wu et al. [32, 33], Wu et al. [34, 35], have shown how ERP adopters can use financial valuation techniques, namely Real Option Analysis (ROA) [26–28, 32–35] and portfolio management models [29–31] when it comes to evaluating investments in large ERP assets under uncertainty.

Both ROA and portfolio management techniques were first developed as decision support approaches in the area of capital investments. In the area of IT, they are deemed ‘promising’ [26] because they let the ERP adopter account for both uncertainties and management flexibilities. In ROA, the concept of ‘real’ means adapting mathematical models used to evaluate financial options to more-tangible investments. Since 1999, most notably by Amran and Kulatilaka [26], this concept has found its way into the area of appraising IT investments, and consequently, ERP investments [27–29, 32–35]. The core of the ROA for ERP assets consists of: (i) the identification and the assessment of optional components in a large ERP project and (ii) the selection and the application of a mathematical model for valuing financial options that serve to quantify the current value of choosing these components for inclusion at a later time. The term ‘optional components’ is used to mean those project parts that can either be pushed ahead or pulled out at a later point in time when new information becomes available to the decision-makers. The option, therefore, is right but not the obligation to spend a budget or put resources on a project. For example, when planning an ERP project with a significant amount of data-mining functionality, it is often possible to first implement a data mart, and then later decide to implement the data warehouse component of the ERP package.

Each of the studies in [26–28, 30–35] has presented the mechanics of a valuation technique, as applied in a specific context, and has clearly demonstrated specific benefits of its application. For example, Wu et al. [32, 33] present a solution to resolve in changing environments those ERP project uncertainties that cannot be predefined. These authors demonstrate the benefit of their approach. In that management can take advantage of favorable changes as they evolve and can evaluate the impact of their actions on the ERP success. Furthermore, Wu et al. [34, 35] use ROA to resolve uncertainties posing risks to the realization of ERP project benefits, Bardhan et al. [29] use option-based models to cope with uncertainty in the prioritization of ERP projects within a large ERP portfolio, Taudes deploys a ROA-based decision-making model to address uncertainties in the valuation of projects aimed at upgrading the existing ERP platforms and choosing among alternative ones [27]. Ranganathan and Brown [28] use a ROA-based model to investigate how three ERP project variables, namely the choice of functional modules to be implemented, the choice of sites where the system will be rolled-out, and the choice of a vendor, impact the overall company’s value of the ERP investment. Based on a study of 116 projects in the U.S., Ranganathan and Brown [28] indicate support for the hypothesis that ERP projects with greater functional scope or greater physical scope (multiple sites) result in positive, higher returns for the ERP adopter.

Empirical studies on the industrial intake of ROA and portfolio management techniques indicate, however, that despite the known advantages of these techniques, their application in practice is still very limited [26]. Amram and Kulatilaka [26] explain that for ERP adopters, using these techniques remains a practical challenge, because of two reasons: (i) the techniques are very data-intensive and very few organizations among the ERP adopters have collected data to feed into the mathematical models, and (ii) they heavily rely on the presence of financial valuation experts specialized in the application of these techniques to IT (or ERP) assets. In reality, when an ERP adopter initiates
a new project, these techniques often turn out to be either irrelevant (as no company-specific data are available to run them) or are prohibitively expensive (as data might be scattered among various project management offices and the company needs to engage a management consulting firm with experts in financial valuation techniques who should work on the raw data from various sources to consolidate it and make sense on how to use it in ROA or portfolio management models).

2.2.2. Related work from the software measurement literature. In the past 5 years, the software measurement community proposed a variety of solutions to uncertainties. Based on our search in the four bibliographic databases, we selected 15 publications which formed this section [5, 36–48]. The starting point of the solution proposals in these publications is the understanding that uncertainty (also termed as ‘epistemic uncertainty’ [49]) results from inaccurate or incomplete information and can be reduced or even eliminated given better models or additional observations. As Aren et al. [39] indicate, to develop a solution to uncertainty one usually devises an axiomatic definition of what uncertainty means and then build a (rather sophisticated) formal theory for the purpose of modeling uncertainty. Aren et al. found that one solution idea on which the software measurement community seemed to agree is to incorporate uncertainty-handling techniques into traditional effort estimation models (e.g., COCOMO II [50]) by using concepts of (i) fuzzy sets theory [51], (ii) probability theory [5, 43], or (iii) evidence theory [52]. For example, instead of using ‘data points’ as inputs into algorithmic models of effort estimation, one could and should consider representing uncertain inputs by using probability distributions. These uncertain inputs are, then, processed by means of some simulation techniques, for example, a Monte Carlo simulation, as applied by Aguitar [48], McDonald et al. [47], Aren et al. [39], and Coleman et al. [41], or a Latin Hypercube simulation, as applied by Garvey [43]. These authors [39, 41, 43, 47, 48] suggest that, for estimation analysts, a recognized advantage of using such an approach is the ability to calculate the implications of uncertainties (by means of the simulations themselves).

Furthermore, proposals to cope with software project uncertainties have been made by Li et al. [42], who model uncertainty by means of a generalized stochastic Petri net and a Markov process algorithm, and by Garvey [43] and Stamelos and Angelis [44], who model uncertainty as a probability distribution and integrate it into a portfolio management process. While Garvey [43] used his approach to estimate the probability distribution of earnings and losses incurred by an organization with respect to its software portfolio, Stamelos and Angelis [44] used their technique to derive confidence intervals for the effort needed for a project portfolio.

Our review of the techniques presented in [5, 36–48] indicates that despite these techniques representing candidates for solutions to consider in resolving uncertainties of ERP project context, their practical application seems problematic for at least three reasons: First, it seems that very few of the literature sources in our review have studied the extent to which the ERP/software phenomenon to be modeled satisfies the assumptions of the underlying technique being used. Having insufficient or no knowledge on the assumptions behind a technique leaves researchers and practitioners with incomplete understanding on when and in which context which technique to use.

Second, if the estimation analyst wants to combine an uncertainty-handling technique and a traditional effort estimation technique for the purpose of ERP project estimation, he or she is confronted with a great deal of inflexibility imposed by the existing effort estimation techniques. Under ‘inflexibility’, we mean that existing models take as inputs pre-defined parameters. For example, earlier research by Stensrud [18] and Stensrud and Myrtveit [19] indicated that traditional effort estimation models consider size to be an one-dimensional concept (e.g., either FP or lines of code). Stensrud [18] warns that this might not be enough in the ERP project realities, which might be better characterized by means of a multi-dimensional size measure.

Third, traditional models are based on an accumulated body of software measurement knowledge from the past 35–40 years [50], whereas effort estimation for ERP is a relatively recent topic and could not take advantage of such a body of knowledge. Consequently, any result in the area of ERP
estimation would be more or less preliminary and few certain conclusions could be drawn from it. We illustrate this point by using an example of the practice of converting software size from FP to lines of codes [38]. (Conversation ratios have been produced since 1985 [38].) For example, Jones [38] makes the observation that ‘For common languages as COBOL many hundreds of projects have now been evaluated by actual counts in terms of both FP and source code size. For some languages, textbooks provide simultaneous implementations of the same algorithms in various languages’. For ERP languages (for example, ABAP/4, the language of SAP), to the best of the author’s knowledge at the time of writing this paper, no study has been published yet and, may be, few programs have been enumerated and results, if any are available at all, might be kept confidential within the ERP software companies. This matter alone makes it very difficult for ERP estimation analysts to determine the validity of any equation that claims to convert a number of FP into a number of lines of code for ERP projects.

In the light of these issues, one could easily imagine that ERP adopters remain reserved toward determining a level of trust in any estimate. Examples of some specific barriers to trust, which researchers [10, 12, 15, 18, 19, 21] have found to be traceable to the six perspectives discussed in this section include: lack of consensus on the objectives of the estimates [15, 18], no known steps to ensure the integrity of the estimation process [12, 18, 19], no historical evidence at the ERP adopter’s site supporting a reliable estimate [10, 21], or the inability to clearly see whether or not estimates are consistent with consultants’ demonstrated accomplishments on other projects in comparable organizations in the same sector [10].

3. THE APPROACH: COMPLEMENTING MULTIPLE TECHNIQUES

Our solution is based on three techniques: (i) the COCOMO II reference model [50] that lets us account for ERP adopter’s specific cost drivers, (ii) the Monte Carlo simulation [47] that lets us approach the cost drivers’ degrees of uncertainty, (iii) the effort-and-deadline-probability-based portfolio management concept [46] that lets us quantify the chance for success with proposed interdependent deadlines for a set of related ERP projects. We chose the combination of (i), (ii), and (iii), because other researchers already experimented with it [45] and found it encouraging. Unlike [45], where the three methods were used complementarily for the purpose of custom software development, we adapt each of the methods to the context of ERP projects at the stage of very early requirements and adopt their joint use therein. We state explicitly that we do not design a new effort estimation model. Instead, we apply a combination of existing approaches in a new setting, namely ERP. We follow [53], in that we consider a project quote to consist of three components, estimated cost, profit, and contingency, and we focus on the models used to estimate cost in particular. We do not cover issues pertaining to profit and contingency. We aim to use the findings from this exploratory study as a basis for more in-depth studies specifically in the area of ERP cost estimation.

3.1. COCOMO II

COCOMO II [50] is one of the best-known algorithmic model for setting budgets and schedules as a basis for planning and control. It comprises:

- five-scale factors, which reflect economies and diseconomies of scale observable in projects of various sizes; that is, these parameters have large influence on big projects and small influence on small projects;
- 17 cost drivers, which serve to adjust initial effort estimations; these are called effort multipliers (EM) and mean parameters that affect effort by the same amount regardless of the project size.

In COCOMO II, the degrees of both the scale factors and the cost drivers vary from extra low, very low, low and nominal to high, very high, and extra high. Suppose that if ERP project
M. DANEVA

stakeholders assign a degree to each scale factor and cost driver, then the estimation of project effort and duration will result from the two equations below:

\[
\text{Effort} = A \times (\text{Size})^E \times \prod_{i=1}^{17} EM_i \tag{1}
\]

and

\[
\text{Time} = C \times (\text{Effort})^F \tag{2}
\]

where \(E\) and \(F\) are calculated via the following two expressions, respectively:

\[
E = B + 0.01 \times \sum_{j=1}^{5} SF_j
\]

and

\[
F = D + 0.2 \times (E - B)
\]

In (1) and (2), \(SF\) stands for the scale factors, and \(EM\) means cost drivers. In (1), size is expressed in lines of code or in FP. The COCOMO II procedure for determining size in FP complies with the ISO functional size measurement standard [54], also referred to as the FPA method of the International Function Point Analysis User Group (IFPUG). When size is in FP, in order to apply Equations (1) and (2), FPs have to be converted to lines of code in the implementation language (e.g., for the SAP ERP package, this language is ABAP/4). COCOMO II does this by means of a backfiring technique [38], which relies on default ratios of FP to line of code conversion. For example, for ABAP/4, the SAP programming language, 1 FP converts into 16 lines of code in this language. (The conversion ratios are updated by and made available to effort estimation analyst at www.spr.com/library/0Langtbl.htm.)

We included COCOMO II in our approach for the following reasons:

First, it offers a balanced set of (i) context characteristics, which usually are a subject of cost communication with the ERP consulting companies participating in a request-for-proposal process and (ii) characteristics that are internal to the ERP-package being adopted (and thus are non-negotiable with ERP service providers).

Second, we found that in ERP project settings, at least three out of the five scale factors, included in COCOMO, are directly related to the joint RE and architecture design activities, and thus raise the role of architects in reducing project costs. COCOMO II allows ERP teams to include in their estimates (i) the maturity level of the ERP-adopting organization, (ii) the extent to which requirements’ and system architecture’s volatility is reduced before ERP configuration, and (iii) the level of team cohesion and stakeholders’ participation. We considered the mapping of these three-scale factors possible because previously published ERP case study research by Markus et al. [54], Holland and Light [55] and Boonstra [56] indicated a linkage between project outcomes and level of maturity in ERP [54, 55], requirements volatility [57], and stakeholders’ involvement [56]. (We refer interested readers to these references for more information.)

Third, the COCOMO II model is in the public domain and it is available for free to companies to use it.

Fourth, the model’s feature that it can use size expressed in FP is important to ERP adopters because they need project estimation at the stage of bidding (or early requirements) and FPs are usually the only sizing data available at that stage.

3.2. The Monte Carlo simulations

The term ‘Monte Carlo’ refers to the field of applied and computational mathematics and denotes a broad family of techniques used to approximate such quantities as integrals and sums of random variables, for which analytic, closed-form formulas are not available because of the form or complexity of the situation [39]. In the area of software effort estimation, a Monte Carlo simulation
The specific step-by-step procedure in which we applied the Monte Carlo simulation technique was NOSTROMO, originally devised by the THAAD Project Office [47].

3.3. The portfolio management concept

We couple the above techniques with a portfolio management concept [34] based on an effort-and-deadline-probability model that allows us to quantify the uncertainty associated with a project estimate. Our goal behind the inclusion of this technique is to respond to the burning issues as raised by Osei-Bryson et al: ‘The lack of good explanatory and predictive models makes it difficult for managers to develop and plan ERP implementation projects with any assurance of success’ [24]. As McConell [58] states ‘a primary goal of any thorough project estimating process should be to not only yield estimated values for these metrics; it should also indicate whether or not these
estimated values satisfy their corresponding project goals within some corresponding specified confidence limits (probabilities of success). We chose this portfolio management model [46] because (i) it is applicable at the stage of requirements or project bidding [46], (ii) its only input requirement is a record of previous projects; although it does require an effort estimate for every project, its need is nothing more sophisticated than a subjective opinion [46]; and (iii) it fits with the ERP adopters’ project realities suggesting that an ERP project is implemented as a portfolio of interdependent subprojects [2, 16, 29, 51]. We make a note that breaking down a complex projects into subprojects and managing it as a portfolio has also been recommended by RAND Corporation [39] and by McConnell [58].

In our approach, each subproject is a piece of functionality (or an ERP module) linked to other pieces (or modules). For example, the Sales and Distribution module in an ERP package is tightly linked with the Accounts Receivable and Profits Center Reporting functionality of the Financial Accounting and Controlling modules. Suppose we have a set of interdependent subprojects, the effort-and-deadline-probability model [46] will yield (i) the probability of portfolio’s success with the proposed deadlines for each subproject in this portfolio, and (ii) a set of new deadlines that will result in a required probability of success. The portfolio success is judged by two conditions applied to any two subprojects $a$ and $b$ for which deadline$_a$ is earlier than deadline$_b$. The conditions are that: (i) subproject $a$ is to be over by deadline$_a$, and (ii) subproject $a$ and subproject $b$ are to be over by deadline$_b$. In other words, the conditions require that all subprojects planned with a deadline before deadline$_a$, should be completed by deadline$_a$, rather than just project $b$. This is the key to the portfolio approach, because uncertainty about completion of project $b$ incorporated uncertainty from all previous projects.

Suppose the ERP adopter engages in total $E$ people in the project and let $d$ be the number of work days it takes from start date to deadline, then the total available resources is $E \times d$. Hence, suppose an ERP portfolio $Y$ is made up of $n$ subprojects, then the success conditions are represented as follows:

$$
\begin{pmatrix}
Y_1 \\
Y_1 + Y_2 \\
\vdots \\
Y_1 + Y_2 + \ldots + Y_n
\end{pmatrix} \leq \begin{pmatrix}
d_1 \\
d_2 \\
\vdots \\
d_n
\end{pmatrix}$$

where $Y_i$ is the estimated effort for subproject $i$ to succeed. We check if, for any $j$, $(j = 1 \ldots n)$ the sum of $Y_1, \ldots, Y_j$ is greater than $E \times d_j$. If this is true, then deadline $d_j$ has failed. Success probabilities result from simulations in which $Y_1, \ldots, Y_n$ is generated from a predetermined probability distribution. If we deem $Y_1, \ldots, Y_n$ is satisfying all conditions, then we say that the portfolio $Y$ succeeds. The portfolio’s probability of success is equal to the ratio of the number of successes in the set $Y$ to the number of trials in the simulation.

### 3.4. The approach applied

Our solution approach starts with the position that at the point in time when we produce the estimate, we can posit a trade-off of probability of success for schedule and other resources [36]. Suppose the ERP adopter’s organization (i) has reasonably characterized the context, (ii) has established some operational variance for the size of the ERP portfolio and the complexity of the subprojects included in it, (iii) has some understanding of the ranges of difficulty in obtaining the right configurations for the system, and (iv) has some calibrated way of processing this information, then it is possible for the organization to simulate what might happen when the ERP adopter runs the project.

Our solution approach consists of eight steps, which are presented in Figure 2. Because we designed our approach with the RE stage in mind, we suggest Unadjusted FP [59] be used as a size estimate. This is consistent with the position of the COCOMO II authors [50, p. 17]. We chose this measure of functional size because (i) it is appropriate to the software artifact being measured.
at the stage of bidding, namely to documents (e.g., business process and data requirements) that are the only project deliverables available at that time [38]; (ii) it is applicable to any ERP package and not to a specific package’s context [60]; (iii) it allows direct measurement of the rate at which requirements change and, thus, may make projects grow larger during implementation; this is possible because both the original requirements and changed requirements will have FP counts; (iv) it has been used in industrial studies for more than 20 years [38]. Clearly, one might think of other functional size measurement standards, for example, COSMIC [61], Mark II [62], NESMA [63], or FISMA [64] as candidates for inclusion in an effort estimation study. However, we preferred the IFPUG standard [59], because (i) the COCOMO II model provides estimation equations based on this IFPUG size, (ii) conversion tables for estimating lines of code from IFPUG FP counts have been proposed and updated for more than 20 years, and (iii) published rules for mapping and applying the IFPUG counting process on ERP business requirements have been existing for more than 10 years [60]. (More discussion on the use of the IFPUG FP method in our approach is provided in Section 4.2.)

Furthermore, to account for uncertainty of the ERP project context, we suggest that the COCOMO II model takes as inputs the probability distributions of the five COCOMO scale factors and 17 cost drivers, instead of using as inputs single values (such as in [4]). This design choice has been recommended by the THAAD Project Office [47] and by the JPL NASA practitioners [65] as well. Deploying the Monte Carlo simulation means to ascribe a particular distribution type to an input variable in a model, get randomly selected values, feed them into the COCOMO II model and, then, see how likely each resulting outcome is. In other words, for each uncertain factor, our approach yields possible effort and duration estimation values. In contrast to COCOMO II, our output is the probability distributions of effort and duration and not the most likely effort and duration (which COCOMO II creates).

The probability distributions are, then, fed into the portfolio management method [46]. To run it, we first formulate a condition for success, as in (3), then we bunch projects into portfolios and we obtain the probability of successfully delivering the projects under time constraints as well under effort constraints.

4. RESEARCH METHOD AND ITS APPLICATION

This paper uses a case study research method [66]. Our case study in which we used the steps from Figure 2 was planned and carried out by following the case study research practices recommended by Yin [66]. The choice of a case study method was motivated by the recommendation of research methodologists in software engineering [67–69]. We make a note that while Yin considers the
distinct advantage of the case study method when a ‘how’ or ‘why’ question is being interrogated about a contemporary set of events over which the researcher has little or no control, the software engineering researchers [67–69] emphasize that case studies are also useful in answering a ‘which is better’ question [68, 69]. The latter is what we are after in this paper. Below, we describe the project settings in which our approach was applied (Section 4.1) along with the details on how we resolved practical difficulties pertinent to each of the three techniques (Section 4.2). We state our expectation of the approach and the results we obtained in Section 4.3.

4.1. Description of the project settings

The solution approach was applied in a setting of a large organization-wide ERP roll-out that included eight functional modules of one ERP package (namely SAP) and covered three locations of a North American telecommunication company. The modules were: Material Management, Sales and Distribution, Service Management, Accounts Payable, Accounts Receivable, Plant Maintenance, Project System, and Asset Management. Our data came from 13 SAP projects implemented in the case study company. The projects were carried out between November 1997 and October 2003. In this period, the author was employed by the case company as a SAP process analyst and was actively involved in the projects. The ERP implementation process model adopted in the context of the projects was the AcceleratedSAP (ASAP) process for implementing SAP solutions [70]. It is a project-specific process, engineered and standardized by SAP, and provided to clients by ASAP-certified consulting partners. The ASAP process has been extensively elaborated in [70]. The process model includes five phases:

1. **Project Preparation**, which is concerned with organizing the executive kick-off meeting and setting up the project organization. It includes training the team, once being composed, drafting a rough project plan, and checking hardware orders.

2. **Business Blueprint** phase, concerned with joined RE and architecture design.

3. **Realization** phase, which includes configuring the baseline system to match 80% of the business processes. We make a note here that the adopter-specific customized system is not configured in the blueprint stage, so that the system at this point (called baseline system) is the basis for the production system, that is, the one that will be delivered and used. Step-wise fine-tuning transforms the baseline system into a production system.

4. **Final Preparation**, concerned with the final system tests (e.g., volume tests, user acceptance tests), user training, data migration to the new system, and production start-up strategy (including the setup of internal audit procedures and end-user support, e.g., internal helpdesk).

5. **Go Live** phase, which is concerned with making sure that the business environment is fully supported.

The practical settings for our 13 projects have been described in detail in [4]. They included the following:

- To manage implementation complexity, each of our projects was broken down into a number of subprojects reflecting the number of components to be configured. For example, the first project had to implement six components and was broken down into six subprojects. This is justified because despite the fact that requirements are collectively presented in a single document, specific requirements pertain to a specific component [4]. Requirements are divided among business process scenarios within the overall project and, thus, facilitate subproject estimates. The total number of our subprojects in which the standard ASAP process was instantiated was 67.

- For each subproject, there was a dedicated team. This is a group of individuals who are assigned to a specific subproject, contribute time to and run the RE cycle for this subproject, and deliver the business requirements document for a specific SAP component.

- Each team consisted of one or two SAP consultants who provided in-depth knowledge in both the ASAP implementation process and the SAP components, and a number of business

representatives, the so-called process owners. There were department managers and subject
matter experts who contributed the necessary line know-how, designed new processes and
operational procedures to be supported by the SAP modules, and provided the project with the
appropriate authority and resources. All process owners had above average level of experience
with IT-projects in their departments and, before starting the projects, attended a 3-h training
session on the ASAP process. Next, we considered our consultants as an even mix of experts,
new hires, and novices. Each expert had at least 5 years of configuration and integration
experience with a specific SAP functional module. Most experts had ASAP RE experience.
Our consulting partners provided evidence that their less experienced staff-members completed
the standard training courses on both the ASAP process and the corresponding SAP modules.
However, none of the consultants had any experience in the telecommunication sector; they
were unaware of the requirements principles in this domain and were supposed to carry out
RE activities under novel and challenging conditions.

- All the teams were supported by a process architect responsible for architecting the solution,
  sharing process knowledge, and consulting on ongoing basis with the teams on SAP reuse,
  process methods, and RE tools. The architect was the only resource the teams shared at all
  ASAP phases.
- Within the final preparation phase, all the teams were supported by a system tester.
- Within the realization phase, all the teams were supported by two ABAP/4 developers who
  programmed the code for those pieces of functionality in the ERP solution, which were added
to the functional modules of the package.
- Within the final preparation stage, all the teams were supported by two data analysts respon-
sible for data conversion.
- The 67 teams (composed of external consultants and business representatives) worked sepa-
  rately and with relatively little communication among them. This allowed us to initially
  consider and include 67 subprojects in our case study.

4.2. Resolving practical problems with the application of the techniques

Because ERP project settings, as those described in Section 4.1, may pose a number of practical
challenges to effort analysts when attempting to use the three techniques included in our approach,
we provide in this section a discussion on the choices and the assumptions we made. Their
implications for validity of the results of our approach are discussed in Section 6. In the present
discussion, we will use the case study settings to illustrate our points with examples.

For each of the 13 projects, we got (i) project size data, (ii) reuse levels, (iii) start and end
dates, and (iv) scale factor and cost driver ratings. Below we will discuss the choices associated
with each of these items.

Functional size: It was measured in terms of unadjusted IFPUG FP [59]. We make a note that
in our particular case study organization, in addition to the reasons mentioned in Section 3.3, this
choice was also welcome due to the matter that the organization already had a IFPUG-compliant
FP measurement practice in place. (The author herself was engaged in translating the IFPUG
counting concept to ERP and this circumstance made possible the collaboration between the author
and an external COCOMO II analyst who was brought into the organization to understand and
assess the cost drivers, and lastly, help locally calibrate the COCOMO II model.) For the purpose
of this case study, the organization did not consider using alternative size measurement standard,
e.g., COSMIC [61] or Mark II [62], because this would have been impractical, given the presence
of already collected IFPUG FP counts.

The FP sizing process is about measuring the functionality by the number of ways the system
interacts with its users. The process includes counting and weighting five data and process logical
components. The data components are the so-called logical internal files and external interface
files, and three transactional components are external inputs, external outputs, and external queries.
For the exact definitions of these counting components, interested readers might consult the IFPUG
counting manual for more information [59]. However, the IFPUG community (e.g., [38, 60]) warns
that the standard FP counting manual falls short in providing details relative to new and emerging
technologies, for example, ERP among others. For the IFPUG counting process to be applied in projects implementing new technologies—like ERP, the standard definitions of the IFPUG data and process counting components must be re-thought in the contexts of these projects and adapted to them [38, 60]. What is unique about the application of the IFPUG standard to ERP is that a set of rules must be established for mapping the data and process counting components of the IFPUG standard onto the components of the data and process requirements models in the ERP business requirements document [60]. These rules say (i) how to identify the boundary of the ERP project to be counted, (ii) how to identify the components to be counted within the boundary of the project, (iii) how to classify these components, and (iv) how to assess the complexity of each component that is counted. Attempts in defining and using such mapping rules have been made since 1998 by the author [60] as well as by other practitioners in ERP size estimation [25, 47, 53, 71–73]. This case study uses the previously published proposal by the author [60]. The step-by-step procedure for counting FP from ERP business process and data requirements is described in [60] and involves three stages: analysis of the process and data components, assignment of complexity values to the components, and calculation of the final FP value. Data collection for FP counting was done by using the paper form designed by the author [60]. (More in detail, the specific challenges of using FP-type measurement model in ERP projects are addressed in [6].)

Furthermore, reuse levels were formed by using a reuse indicator that included reused requirements as a percentage of total requirements delivered [60]. The exact calculation rules for counting FP and reuse indicators (based on FP) are presented in [60].

COCOMO II. For the COCOMO II model to be useful in an ERP setting, an organization should first calibrate it locally. This is because Equations (1) and (2) reflect the experience and data of the COCOMO II developers and their calibration is based on 161 projects from aerospace, commercial and Federally Funded Research & Development Center’s projects [50]. ERP projects are not included in this calibration exercise and, to ERP adopter, this means applying local calibration techniques [50] and using their own project data, so that they get equations that best fit the company environment. Only then, it is safe and reasonable to deploy COCOMO II in their own settings. (As recommended in [50], ‘an organization, using COCOMO, calibrate it using their own data for them to increase model accuracy and produce a local optimal estimate for similar type projects’.)

In this case study, the company lets a COCOMO II expert calibrate the model to reflect local circumstances that may affect its output values. The effort multipliers $A$, $B$, and $EM$ in Equations (1) and (2) and the scale factors $SF$ were calibrated by using (i) the linear regression approach [50] and (ii) ERP effort data collected from the first eight ERP projects that were implemented. Given this number of projects, it meant that we met the recommendation for local calibration that at least five projects be used for calibrating the multiplication $A$ and at least 10 projects be used to calibrate the cost drivers and scale factors [50]. The data needed for calibration purpose were collected online and in a project reporting database.

Our choice of which parameters to calibrate is consistent with the recommendations made by the COCOMO II developers. For example, the decision to calibrate the multiplicator $A$ was based on the organization’s desire to reflect two local circumstances in the model:

- that the life cycle activities in COCOMO II are different from the ones in the ASAP model (described in Section 5.1), and
- that the definitions used by COCOMO deferred from the ones used in the ERP-adopting organization. To illustrate a difference in the meanings of terms between COCOMO II [50] and the case study organization, we use the definition of person month: COCOMO defines one person month to equal 152 work hours, whereas in the case study organization 1 person month was 154 hours.

To those organizations who want to consider in the model these two organizational circumstances (that is, use of local process models and definitions), the COCOMO II authors suggest them to calibrate the multiplicator $A$.

Furthermore, the ratings of the COCOMO II cost drivers and scale factors include interpretation and this mandated to take extra care on our side to ensure consistency. As the organization wanted
local rating criteria for the cost drivers and the scale factors, they elaborated and used formal, objective rules for rating ASAP implementation process in compliance with the company’s Human Resource policies.

We are aware of the calibration issues related to COCOMO II, for example, the ones discussed in [74]. However, we do not focus on them in this paper, because this is not the scope of the present research. (We, however, acknowledge that the calibration of COCOMO II in ERP settings provides fruitful avenues for future research. For more insights into the COCOMO II local calibration issues, readers could refer to [74].)

**FP-to-line-of-code conversion:** This case study organization used the average number of lines of code (ALC) in the SAP particular language, ABAP/4, which are required to implement 1 FP. (According to [38], 1 FP equals to 16 ALC.) The estimated code size was computed as follows:

\[
\text{Code Size} = \text{ALC} \times \text{UFP}
\]

We are conscious about that the FP-to-lines-of-code conversion might be fraught with pitfalls and, hence, might pose a threat to validity. Jones [38], who created and now updates the conversation tables, recommended by the COCOMO II authors [50], acknowledges that the relationship between FP and lines of code is much more complex than backfiring gives credit, which means, in turn, that Equation (4) might be grossly inaccurate. For example, in (6), a linear relationship is assumed, but if the system to be delivered is being developed in a new and very different environment (even when using the same language) that assumption might no longer be justified [38]. A more complete list of the inherent uncertainties with the backfiring method is provided by Dekkers and Gunter [75]. In this case study we take the position of Mah [76] and Dekkers and Gunter [75] that transitioning from one size metric to another is useful, as long as the organization understands ‘the purposefulness of such an endeavor’. In this case study, the purpose of conversion is to make an approximate estimate without undue effort. We follow the recommendations by Dekkers and Gunter [75] that the published conversion ratio should be viewed ‘as no more than indicative’ and its use makes sense for rough calculations when large sample size ‘will even out the discrepancies between types of applications on a portfolio-wide basis’. In our case study, we target an early (and therefore ‘rough’) estimation and have a relatively large sample of projects treated as a portfolio, which led us to the choice to use the conversion equation (6). The issues surrounding the conversion from FP to lines of code in ABAP/4 form part of the discussion on validity in Section 6.

**Monte Carlo simulation:** Because we had the ratings of the cost drivers and scale factors only and no knowledge about the uncertainty of the ratings, we assigned to each factor its distribution type and its parameters of probability distribution (namely center, min, and max) based on previously published experiences and recommendations by other authors [36, 45, 47]. For example, this case study used McDonnald’s [47] default ‘high’ levels of uncertainty associated with the ratings of the RESL, DATA, ACAP, and PCAP cost drivers [50]. (In Section 5, Table III presents descriptions of the COCOMO II cost drivers. For more detailed definitions of the cost drivers, we refer readers to Reference [50].) The level of uncertainty determines, in turn, the distribution type to be assigned to each cost driver: normal, triangular, and uniform for low, medium, and high uncertainty, respectively.

We also opted to use a lognormal distribution for functional size, which was motivated by the observations of Chulani et al. [77]. These researchers investigated the size distribution and indicated that its skew is positive and that log(size) is likely to be a normal distribution.

With this input data (namely the COCOMO II factors and uncertainty values), we run Monte Carlo simulations that gave us samples of (i) effort, expressed in person-month and (ii) time, expressed in months. Generally, a Monte Carlo simulation consists of many, often thousands of, trials, each of which is an experiment where we supply numerical values for input variables, evaluate the model to compute numerical values for outcomes of interest, and collect these values for later analysis. In this case study, we used 10,000 trials and generated the samples of effort and time, as presented in Figures 3 and 4, respectively. In these histograms, the Y-dimension shows
the frequency with which a value was observed in the sample of 10000 trials. The X-dimension shows the value range. Because the average subproject involved four professionals (two business users, one external consultant, and one internal IS team member), we adopted the assumption for \( E \) to be 4.

Based on the observation that COCOMO II provides time estimation as in (2), we formulated the following condition for portfolio management in terms of time constraints:

\[
\begin{pmatrix}
T_1 \\
T_1 + T_2 \\
\vdots \\
T_1 + T_2 + \cdots + T_n
\end{pmatrix} \leq \begin{pmatrix}
m_1 \\
m_2 \\
\vdots \\
m_n
\end{pmatrix} \tag{5}
\]

where \( T_i \) is the ERP implementation time in months for subproject \( i \). In this condition, we did not include the number of people \( E \), because COCOMO II assumed an average number of project staff [50], which was accounted in (2). Furthermore, as recommended in [45], we attempted to improve the chances for portfolio success by adjusting the cost drivers and scale factors. Hence, we adopted the assumption that for projects with two different ratings for the same factor, the probability of success for each project will also be different. Finally, our case study plan included assessment of how much the probability of success increased when treating ERP projects as a portfolio. We expected that the subprojects with high uncertainty ratings would benefit more from portfolio management than the projects with low uncertainty ratings would do.
5. RESULTS

This section reports on the results with respect to: (i) what we observe when adjusting COCOMO II cost drivers and (ii) what we learnt from the probability of success of highly uncertain projects when managing them as a portfolio.

To understand how cost drivers and scale factors make a difference in terms of project success, for each one of them we constructed two portfolios: the first one had this driver/cost factor rated ‘very high’ for all projects and the second portfolio had it rated ‘very low’ for all projects. Given these two portfolios, we ran the portfolio management process. The results of using the two portfolios with respect to each cost driver are presented in Table II. (For reader’s convenience, after the results in Table II, we provide Table III to explain the meanings of the acronyms used for the cost factors in [50]). In Table II, for each of the 17 COCOMO II cost drivers, we present results in two rows: the one showing the probability of success under effort constraints and under time constraints, when the factor is rated ‘very low’ and the second row showing these probability percentages when the factor is rated ‘very high’. For example, we found that when selective reuse [60] was practiced in ERP projects, the probability of success was higher under both time and

<table>
<thead>
<tr>
<th>Cost driver/scale factor rating</th>
<th>Probability of success</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Under effort constraints (%)</td>
</tr>
<tr>
<td>DATA: Very low</td>
<td>88.72</td>
</tr>
<tr>
<td>DATA: Very high</td>
<td>76.91</td>
</tr>
<tr>
<td>CPLX: Very low</td>
<td>93.01</td>
</tr>
<tr>
<td>CPLX: Very high</td>
<td>59.11</td>
</tr>
<tr>
<td>TIME: Nominal*</td>
<td>86.00</td>
</tr>
<tr>
<td>TIME: Very high</td>
<td>86.91</td>
</tr>
<tr>
<td>STOR: Nominal†</td>
<td>82.00</td>
</tr>
<tr>
<td>STOR: Very high</td>
<td>83.78</td>
</tr>
<tr>
<td>RUSE: Very low</td>
<td>68.78</td>
</tr>
<tr>
<td>RUSE: Very high</td>
<td>96.87</td>
</tr>
<tr>
<td>DOCU: Very low</td>
<td>86.51</td>
</tr>
<tr>
<td>DOCU: Very high</td>
<td>61.23</td>
</tr>
<tr>
<td>PVOL: Very low</td>
<td>96.86</td>
</tr>
<tr>
<td>PVOL: Very high</td>
<td>71.21</td>
</tr>
<tr>
<td>SCED: Very low</td>
<td>91.99</td>
</tr>
<tr>
<td>SCED: Very high</td>
<td>76.29</td>
</tr>
<tr>
<td>RELY: Very low</td>
<td>78.99</td>
</tr>
<tr>
<td>RELY: Very high</td>
<td>61.91</td>
</tr>
<tr>
<td>TOOL: Very low</td>
<td>49.27</td>
</tr>
<tr>
<td>TOOL: Very high</td>
<td>98.01</td>
</tr>
<tr>
<td>APEX: Very low</td>
<td>56.31</td>
</tr>
<tr>
<td>APEX: Very high</td>
<td>97.45</td>
</tr>
<tr>
<td>ACAP: Very low</td>
<td>67.99</td>
</tr>
<tr>
<td>ACAP: Very high</td>
<td>91.98</td>
</tr>
<tr>
<td>PCAP: Very low</td>
<td>56.99</td>
</tr>
<tr>
<td>PCAP: Very high</td>
<td>89.99</td>
</tr>
<tr>
<td>PLEX: Very low</td>
<td>64.56</td>
</tr>
<tr>
<td>PLEX: Very high</td>
<td>69.51</td>
</tr>
<tr>
<td>LTEX: Very low</td>
<td>71.50</td>
</tr>
<tr>
<td>LTEX: Very high</td>
<td>91.99</td>
</tr>
<tr>
<td>PCON: Very low</td>
<td>61.09</td>
</tr>
<tr>
<td>PCON: Very high</td>
<td>96.59</td>
</tr>
<tr>
<td>SITE: Very low</td>
<td>69.26</td>
</tr>
<tr>
<td>SITE: Very high</td>
<td>86.00</td>
</tr>
</tbody>
</table>

*The rating of the TIME cost driver ranges from ‘nominal’ to ‘extra high’.
†The rating of the STOR cost driver ranges from ‘nominal’ to ‘extra high’.
Table III. Cost drivers and descriptions as provided in [50].

<table>
<thead>
<tr>
<th>Cost driver acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>Database size</td>
</tr>
<tr>
<td>CPLX</td>
<td>Solution complexity</td>
</tr>
<tr>
<td>TIME</td>
<td>Execution time constraint</td>
</tr>
<tr>
<td>STOR</td>
<td>Main storage constraint</td>
</tr>
<tr>
<td>RUSE</td>
<td>Required level of reuse</td>
</tr>
<tr>
<td>DOCU</td>
<td>Documentation match to life-cycle needs</td>
</tr>
<tr>
<td>PVOL</td>
<td>Platform volatility</td>
</tr>
<tr>
<td>SCED</td>
<td>Scheduling factor</td>
</tr>
<tr>
<td>RELY</td>
<td>Required reliability</td>
</tr>
<tr>
<td>TOOL</td>
<td>Use of ERP implementation tools</td>
</tr>
<tr>
<td>APEX</td>
<td>Application experience</td>
</tr>
<tr>
<td>ACAP</td>
<td>Analyst capability</td>
</tr>
<tr>
<td>PCAP</td>
<td>Configuration Specialist/Programmer capability</td>
</tr>
<tr>
<td>PLEX</td>
<td>Platform experience</td>
</tr>
<tr>
<td>LTEX</td>
<td>ERP-specific language and workbench experience</td>
</tr>
<tr>
<td>PCON</td>
<td>Personnel continuity</td>
</tr>
<tr>
<td>SITE</td>
<td>Multisite implementation</td>
</tr>
</tbody>
</table>

Table IV. Increase in probability of success for low and high uncertain projects under effort constraints.

<table>
<thead>
<tr>
<th>Uncertainty level</th>
<th>Probability of success</th>
<th>Ratio of increase (a)/(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low uncertainty</td>
<td>93.78%</td>
<td>98.81%</td>
</tr>
<tr>
<td>High uncertainty</td>
<td>84.31%</td>
<td>97.76%</td>
</tr>
</tbody>
</table>

Table V. Increase in probability of success for low and high uncertain projects under time constraints.

<table>
<thead>
<tr>
<th>Uncertainty level</th>
<th>Probability of success</th>
<th>Ratio of increase (a)/(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low uncertainty</td>
<td>15.76%</td>
<td>87.52%</td>
</tr>
<tr>
<td>High uncertainty</td>
<td>8.31%</td>
<td>75.91%</td>
</tr>
</tbody>
</table>

effort constraints. For the purpose of illustrating this point, we report on the results (see Table II) yielded when constructing two portfolios of subprojects, namely the first one with the factor of RUSE (meaning reuse) rated as very high for all subprojects and the second one with RUSE rated very low for all subprojects. We make two notes: First, that low level of reuse in an ERP project indicates massive customization of the standard components and that a high level of reuse indicates limited customization [60]. Second, we ruled out the rating ‘extremely high’ as that high levels of reuse are relatively rarely to be observed in an ERP project context [57, 60, 78]. Table II suggests that when a project is composed of subprojects all of which have RUSE rated very high, the probability of success is greater under both time and effort constraints.

We observed that 13 out of the 17 factors from the COCOMO II model can be adjusted in a way that maximizes the probability of success. These 13 factors are: DATA, CPLX, RUSE, DOCU, PVOL, ACAP, PCAP, PCON, APEX, LTEX, use TOOL, SITE, and SCED. In this case study, our results are inconclusive with respect to the ability to adjust 4 out of the 17 cost factors. These are: STOR, TIME, RELY, and PLEX. We also observed that the scale factors do not seem to be adjusted in a way that maximizes the chance of project success.

Regarding our second group of results, our observations suggest that bundling ERP projects as a portfolio had the advantage over managing projects separately in terms of ability to explicitly and systematically approach uncertainty. We compared the probability of success for projects under effort constraints and for projects under time constraints, respectively (Tables IV and V). They
indicate that the probabilities of success for projects with high uncertainty ratings are greater when those projects are managed as a portfolio.

6. DISCUSSION ON THE IMPLICATIONS FOR RESEARCH AND PRACTICE

The case study provided evidence that led us to conclude the following:

(i) when managed as a portfolio, highly uncertain ERP projects have a greater chance to succeed under time and under effort constraints,
(ii) subprojects with high uncertainty ratings would have greater advantages from portfolio management than projects with low uncertainty ratings would do,
(iii) it is possible to adjust cost drivers so that one increases the probability of success for highly uncertain ERP projects, which a company might have to implement. We have also shown that 13 out of the 17 COCOMO II cost drivers can be adjusted to increase the chances for success.

With respects to (i) and (ii), our results agree with the observation by Jiamthubhugsin and Sutivong [45], who experimented with the portfolio management method in the context of custom projects. Though, we must acknowledge (i) that we have preliminary results only and (ii) that related validity concerns [69] remain our most important issue. We are currently carrying out a replication study in a financial services company to test our approach. The results will serve to properly evaluate its validity and come up with an improved version of our method.

This case study showed that, from the perspective of an ERP adopter, good project management goes far beyond accepting the consultant-imposed estimates as targets and technically implementing the system. Our case study results show that there are project context characteristics, which ERP project managers can change, so that their actions would cause a desired change in an important project outcome. This gives us a ground to say that if a project manager is mainly focused on technicalities of rolling out the system in his/her business unit and ignores the context factors, he or she may miss on an important opportunity to leverage his/her knowledge of the organization in order to act in a way increasing the chances for project success. Becoming aware of the diverse ways in which the project manager can act favorably to the project outcome may advance the project as a whole. We believe that there is a direction for future research to turn our approach to a comprehensive ERP project analysis. In this respect, replication case studies need to be carried out in terms of linking project manager’s actions to ERP project success and failure in order to understand how the uncertainty perspectives (described in Section 2.1.) explain failure and success of ERP projects. The role of project managers of ERP adopter organizations in resolving uncertainties also warrants more attention since their role seems to be crucial throughout the project.

The results in the case study show that an increased understanding of ERP efforts is not necessarily obtained by using brand-new tools and project-specific models. Instead, there are some alternative working methods available. By complementing them, our understanding can be improved.

Another theme raised by this case study is the changing perception about ERP project uncertainties. The authors in the software measurement literature, included in the paper [5, 36–45, 47–49], seem to converge on that the uncertainty of cost drivers in project estimation is considered as a challenge and is treated as such. Our case study results suggest that it might make sense to revise this position and consider uncertainty in early ERP projects as an asset and a resource to project managers. We have also found that 13 out of the 17 COCOMO II cost drivers can be adjusted, so as to increase the chances for success. The case study revealed that project managers can react by influencing the project success, for example, by altering the context regarding these 13 cost drivers in ways that are more consistent with the ERP-adopting organization’s interests (as opposed to the interests of the consulting partners).
7. EVALUATION OF VALIDITY CONCERNS

We explicitly consider this study as preliminary and a first step only toward a better understanding of the major phenomena that cause uncertainty in ERP effort estimation. To this end, we could only say that we need to carry out a few replication studies so that the findings of this study can be consolidated and transformed into recommendations to ERP project managers. As per the recommendation of research methodologists [66, 69], we did an early assessment of the following validity [69] threats.

First, the major threat to external validity arises from the fact that the company’s projects might not be representative of the entire population of ERP adopters. We, however, believe that our project context is typical for the telecommunication companies in North America: we judge these settings typical because they seemed common for all SAP adopting organizations who were members of the American SAP Telecommunications User Group (ASUG). The ASUG meets on regular basis to discuss project issues and suggests service sector-specific functionality features to the vendor for inclusion in future releases. The SAP components our case company implemented are the ones that other ASUG companies have in place to automate their non-core processes (accounting, inventory, sales and distribution, cell site maintenance).

Second, we make a note that COCOMO II (as any other model of this family) is a mathematical representation of ‘idealized’ real-life project relationship only. This is to acknowledge the fact that the 23 cost drivers and cost factors can lead to strong collinearity, heteroscedasticity, and highly variable prediction accuracy [79]. In this case study, for the sake of simplicity and pragmatism, we did not investigate the possible ways in which dependencies among cost drivers of different natures might impact our analysis. We, however, consider it as an important issue and plan to research it in the future. To this end, we could only offer an advice to those practitioners and researchers interested in replicating our approach that they should be conscious about this particular under-researched feature of the underlying techniques being used.

Third, we acknowledge that a threat to validity is posed due to two fundamental difficulties specific to the use of all algorithmic models (such as COCOMO II):

1. Conversion of FP to lines of code is easy to carry out, but is often inaccurate [38]. In ERP, inaccuracies may arise also because the code size is affected by package configuration and design decisions, which have not been made so early in the requirements stage. For example, the SAP package uses a commercial database (e.g., Oracle or Sybase) and, thus, the code size might be small but additional efforts might be needed to overcome the performance limitations of the database product.

2. The estimates of the context characteristics contributing to E and EM are subjective (see Equation (1)). We acknowledge that variation may exist from a team member to a team member, depending on the depth and the breadth of the team members’ practice and experience. We, however, believe that the threat to validity in this respect is reduced because we used rules for determining the rates of the cost drivers specifically elaborated for the specific organization and the specific project type. In this case study, the external COCOMO II expert made sure that these rules are consistently applied in the local calibration process of the COCOMO II model.

Fourth, we must make a note that FP might not be the best way to characterize the size of ERP projects. Stensrud [18] hypothesizes that ERP projects might need a multi-dimensional measure of size, and if that is the case, then FP would not be a good enough measure for the purpose of ERP, as it is an one-dimensional measure. However, Stensrud calls his research ‘non-empirical’ [18], meaning that it provides nothing more than a sound analytical argument that still remains to be backed up by empirical studies in real-life projects. Thus in practice, to ERP adopters, the dilemma is either to go with an ISO-standardized functional size measure (e.g., IFPUG FP [80]), which is one-dimensional and, therefore, might not be enough to characterize ERP, or to devise a multi-dimensional measure of size (as the one presented by Stensrud [18]), which is not validated and might open a number of new issues when used in effort estimation models. In our case study organization, we adopted
the IFPUG standard [59], because we did not have any argument, based on solid empirical data, pointing to why we should rule it out. In the case company, those professionals involved in the estimation of ERP projects supported the position that the risk of having inaccuracy due to the use of IFPUG is more bearable than the risk of introducing in the organization a size measurement model that was not validated. The analysts believed that the organizations would not venture out to design their own measure of size as they had neither the expertise nor the time to do this in a disciplined way.

Fifth, a threat to validity might be posed because of the use of the conversion ratio, ALC, in Equation (6), which helps to transform the size in FP into lines of code in the COCOMO II model. As Mah [76] suggests it is always a good idea to use local data to identify an organization-specific conversion ratio. In the study, determining a company-specific ratio was not a viable option, as we did not have the data and the resources this would need. Instead, we used the most recent table by Jones [38], in which a range of possible values (that is, the min and the max number of lines of code, corresponding to one FP) is assigned to a programming language. The mode value in this range is also provided, which we used in this study. The decision to use the mode value was made after a discussion with three IFPUG members who also were involved in the development of the conversion ratios in the table.

Sixth, we make a note that there is a justified validity concern when our approach is applied in an ERP setting where multiple languages are used to develop different parts of the ERP solution. However, in the present case study, we believe that this threat was no consideration, because all custom-made extensions to the standard SAP system were written in the SAP language ABAP/4.

Seventh, when constructing the portfolio, the author based her choice of ‘very low/very high’ ratings on her own experience in implementing ERP. While for some drivers, as reuse, the author did research on what reuse levels are achievable in an ERP project [60], for others the author set up the ratings in a way that, clearly, could be subjective. However, this design choice was the only possible way to go, given the fact that, to the best of our knowledge, there is no published research on the COCOMO II factor ratings that are more common in the ERP context. We plan, in the future, to research the topic of economies and diseconomies of scale in ERP projects, hoping that new knowledge will help to refine our approach.

Eighth, we deployed complementary three models of three types and are well aware of other possibly useful techniques by each type. Instead of COCOMO II, one may decide to use COSYSMO [81] or COPLIMO [82]. The first one is about estimating large system-of-systems projects and it might fit with those cross-organizational ERP settings in which an overall ERP system is composed of modules of different packages (e.g., SAP, Oracle, and PeopleSoft) [6]. The second refers to the estimation of product line software and focuses on modeling the portions of the software that involve product-specific newly built software, fully reused black-box product line components, and product line components that are reused with adaptation. This approach to reuse seems to converge with the definitions of ERP reuse concepts in [60].

Furthermore, instead of the Monte Carlo simulations, the Latin Hypercube technique could be used [41, 43]. Instead of the portfolio management model by Fewster and Mendes [46], the portfolio approach of Verhoef [31] might be good candidates for inclusion. In the future, we are interested in investigating whether different modeling choices sustain our results or limit the validity of our findings to the subset of the analyzed models. We make a note, however, that when it comes down to the practical applicability of the various combinations of techniques by these three types, it all will depend on the availability of data by certain types at ERP adopter’s sites. Hence, the choice of techniques should represent an acceptable balance between data-intensiveness and usefulness.

As pointed in Section 2, some techniques are more data-intensive than others and, therefore, we expect that their usefulness will vary. We expect that those techniques that are more data-intensive will be of more limited use, as very few ERP organizations have the practice of disciplined project data collection and reporting.

Finally, we would like to note that we do not plan to run a study of competing software prediction models to answer the question of ‘which prediction model is the best?’ This decision is justified by the findings of other researchers [83] who studied the research procedures used as a basis of model
comparison. These authors found that current research procedures need to be more reliable so that researchers can gain confidence in the conclusions of comparative studies of software prediction models. Instead, we plan to concentrate on the question of what combination of techniques could fit best to which ERP context. We are interested in improving our understanding of the effect of assumptions commonly made in uncertainty quantification processes. We would like to compare the possible ways of representing uncertainty for ERP stakeholders. The latter two items form our immediate research agenda.

8. CONCLUSIONS

This empirical study confirms our belief that a joint application of multiple techniques has the potential to aid ERP project estimation analysts. The paper provided researchers and practitioners an alternative look at the ERP cost estimation problem. The purpose of this paper was not to undermine the existing approaches but rather to suggest that more multi-technique-based approaches should be proposed for the ERP context. In our solution, certain steps are meant to overcome the weaknesses of the three techniques being used. We assume, however, that practitioners and researchers can find other means to improve the modeling of uncertain ratings of cost drivers and scale factors. An important condition of using our approach is to understand the intent of the application of each technique, so that effort analysts can be sure that they are overcoming the weakness of each technique in other ways.

In this case study we have demonstrated that the complementary use of the Monte Carlo simulation, a portfolio management method, and a parametric empirical model (COCOMO II) can help to counterpart the uncertainty in early ERP effort estimation based on business requirements and to ensure that setbacks in some ERP implementations are balanced by gains in others. The targeted effect was to systematically cope with two aspects inherent to ERP project contexts: (i) lack of ERP-adopter’s specific historical information about the context and (ii) strong bias by outsourcing partners and ERP consultants in cost estimation. We found this approach to be one good alternative to ERP-adopters as they no longer have to live with whatever estimates are given to them by ERP consultants.

ACKNOWLEDGEMENTS

This paper was partially funded by the NWO projects CARES and QuadREAD. The author thanks the anonymous reviewers for their insightful comments that improved the paper. The author thanks Alain Abran, Nelly Condori-Fernandez, and Carol Dekkers for the discussions on functional size measurement standards.

REFERENCES


63. NESMA. NESMA functional size measurement method compliant to ISO/IEC 24570, 2006. Available at: www.nesma.nl.
76. Mah M. Determining your own function point and lines of code proportions—Three things to watch out for, IT metrics strategies, *Cutters Consortium*, November 2000.


**AUTHOR’S BIOGRAPHY**

**Maya Daneva** is currently an assistant professor with the Information Systems department at the University of Twente, where she leads a research program on ERP requirements engineering and requirements-based ERP project estimation. She serves as the empirical research liaison to the Dutch IT industry and the Dutch Function Point User Group (NESMA). Prior to this, she was a business process analyst in the Architecture Group of TELUS Corporation in Toronto, Canada’s second largest telecommunication company, where she consulted on ERP requirement engineering processes, architecture reuse, and function-point counting methods for SAP projects. In 1994–1995, Maya was a postdoctoral research fellow at the Institute of Wirtschaftsinformatik, at the University of Saarbruecken, Germany, where she worked on benchmarking approaches to improve business process models. She has authored more than 70 research and experience papers in journals and in conference proceedings with Springer, IEEE and ACM.