Information and Software Technology 59 (2015) 136–148

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

# Analyzing impact of experience curve on ROI in the software product line adoption process

Eray Tüzün [a,*], Bedir Tekinerdogan [b]

[a] Information and Security Technologies Division, Havelsan A.Ş., Ankara, Turkey
[b] Computer Engineering Department, Bilkent University, Ankara, Turkey

ARTICLE INFO

*Article history:*
Received 22 November 2013
Received in revised form 16 July 2014
Accepted 18 September 2014
Available online 16 October 2014

*Keywords:*
Experience curve
Learning curve
Software product line engineering
Cost models
Productivity
Software reuse

ABSTRACT

*Context:* *Experience curve* is a well-known concept in management and education science, which explains the phenomenon of increased worker efficiency with repetitive production of a good or service.
*Objective:* We aim to analyze the impact of the experience curve effect on the Return on Investment (ROI) in the software product line engineering (SPLE) process.
*Method:* We first present the results of a systematic literature review (SLR) to explicitly depict the studies that have considered the impact of experience curve effect on software development in general. Subsequently, based on the results of the SLR, the experience curve effect models in the literature, and the SPLE cost models, we define an approach for extending the cost models with the experience curve effect. Finally, we discuss the application of the refined cost models in a real industrial context.
*Results:* The SLR resulted in 15 primary studies which confirm the impact of experience curve effect on software development in general but the experience curve effect in the adoption of SPLE got less attention. The analytical discussion of the cost models and the application of the refined SPLE cost models in the industrial context showed a clear impact of the experience curve effect on the time-to-market, cost of development and ROI in the SPLE adoption process.
*Conclusions:* The proposed analysis with the newly defined cost models for SPLE adoption provides a more precise analysis tool for the management, and as such helps to support a better decision making.

## 1. Introduction

Software reuse has been an important goal in the history of software engineering [1]. Early reuse approaches such as abstract data types, module-based programming, component-based software development, reusable libraries and design patterns, can be basically categorized as small-scale reuse [1]. On the other hand, software product line engineering (SPLE) aims to provide proactive, pre-planned reuse at a large granularity to develop applications from a core asset base [2]. A product line is defined as a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [2]. The benefits for adopting a product line approach have been analyzed and discussed before by several authors [2–5], and usually include developing products more efficiently, get them to the market faster to stay competitive and produce with higher quality [5].

The core asset base in SPLE is developed through a domain engineering process and requires an upfront investment even before a single product has been developed. Despite the upfront investment it is believed that SPLE will result in a *Return on Investment* (*ROI*), in which the development cost and time-to-market value is lower than in case a single system development would have been used. In parallel with this assumption, it is generally acknowledged that transitioning to a product line engineering approach needs to be performed carefully to avoid failures and mitigate risks that are inherent to the adoption of the product line engineering. Hence, the product line engineering community has proposed different transition strategies that aim to support the transition process and as such help to define a proper product line engineering approach for the organization. Hereby, for analyzing the ROI in the adoption of the SPLE process, various cost models have been developed that typically focus on the cost for preparing the organization, the cost for developing the asset base, the cost of unique development and the cost for reusing assets. The result of this analysis process is typically used to provide a decision on whether to

* Corresponding author.
*E-mail addresses:* etuzun@havelsan.com.tr (E. Tüzün), bedir@cs.bilkent.edu.tr (B. Tekinerdogan).

http://dx.doi.org/10.1016/j.infsof.2014.09.008
0950-5849/© 2014 Elsevier B.V. All rights reserved.

adopt an SPLE approach, and if so, which transition strategy to select.

A close study to the cost models and the transition strategy approaches shows that the experience gained by the developers in the development of the products is not explicitly considered in the corresponding cost models. From practice, it is known that most tasks can be carried out faster with increased experience. A more experienced developer who has worked on similar tasks for a long time will be able to develop products faster and on its turn the cost to develop the subsequent products will be reduced. The experience of the developer as such can have a direct impact on the success or failure of the adoption of SPLE. This observation implies that the experience effect should also be explicitly considered in the cost models adopted in SPLE transition strategies.

In fact the impact of experience has been widely discussed in different disciplines including management science, education, construction and manufacturing, in which the notion of *experience curve* has been introduced to analyze and measure productivity. In this context, experience curve explains the phenomenon of increased worker efficiency and improved organizational performance with repetitive production of a good or service [6].

In this paper we aim to analyze the impact of the experience curve effect on the ROI in the SPLE process. For this, we first investigate the impact of experience curve in software engineering in general, by adopting the results of the systematic literature review on experience curve in software development practices. The systematic literature review resulted in total 15 primary studies that confirm the impact of the experience curve in software development projects. In alignment with the results of the identified primary studies we study the impact of experience curve on the adoption of software product line engineering. As stated before, an important metric for analyzing the impact of the SPLE adoption is the ROI for which various cost models have been provided in the SPLE community [7,8]. It appears that none of the cost models as published in the SPLE literature consider the impact of experience curve effect explicitly. Hence, in this paper we provide an analysis approach for integrating the experience curve effect within the current cost models for analyzing ROI of the adoption of SPLE. With the adapted cost models we aim to provide a more precise analysis for the adoption of the SPLE and the expected ROI. The presented analysis shows new insight in comparing single system development with product line engineering that can impact the decision for adopting SPLE or not. We illustrate the application of the analysis for predefined SPLE adoption scenarios. Further we discuss the result of the application of the approach within a real industrial context of Havelsan.

The remainder of the paper is organized as follows. In Section 2 we describe the background on the experience curve effect. Section 3 presents the systematic literature review on the impact of experience curve in software engineering. Section 4 presents the SPLE cost models and the proposed cost models that integrate the experience curve effect. Section 5 shows the analysis using the refined cost models. Section 6, discusses the adoption of the analysis approach for a real industry project. Section 7 provides the discussion and the general practical implications of the approach. Section 8 presents the related work and finally Section 9 concludes the paper.

## 2. Experience curve effect

From practice we can state that the more frequent a task is performed, the less time is required on each subsequent iteration. This increase in efficiency is inherently based on the *learning effect*; after each iteration more will be learned on applying the task and likewise the subsequent iteration will take less time. This phenomenon was first observed by the 19th century German psychologist Ebbinghaus [9] who investigated the difficulty of memorizing nonsense syllables, and recorded the success over a number of trials. According to Ebbinghaus the sharpest increase of learning occurs after the first trial, and gradually evens out after subsequent trials. The corresponding learning curve is a graphical representation of the increase of learning with experience. Hereby, the horizontal axis represents the experience either directly as time spent on the activity, or it can be related to time and define the number of trials. The vertical axis represents the learning or proficiency. Depending on the adopted metric it can either be increasing (e.g. the score in a test), or decreasing (the time to complete a test).

Initially, the learning curve concept related the time required to perform a task to the number of times the task has been performed. Soon it was observed that the learning curve is quite general and can be applied to other domains than learning and psychology. In 1936, Theodore Paul Wright proposed a mathematical model of the learning curve and described the effect of learning on production costs in the aircraft industry [10]. In the corresponding studies, a systematic decline in the labor hours required to produce an airplane was observed. Every time the total aircraft production doubled, the required labor time decreased by 10–15 percent. In 1968, Bruce Henderson of the Boston Consulting Group (BCG) further adapted the model of Wright and proposed the so-called *experience curve*. In the experience curve, unit cost is plotted against total production. The experience curve is currently a broad concept that is applied in different domains. The main reason for the reduced cost is often attributed to the faster development of the tasks.

The mathematical formula for experience curve is relatively simple and is defined as a power law function:

$$C_n = C_1 n^e \tag{1}$$

where

- $C_1$ is the cost of the first unit of production,
- $C_n$ is the cost of the $n$th unit of production,
- $n$ is the cumulative volume of production,
- $e = \frac{\log\alpha}{\log 2}$ = The learning index,
- $\alpha$ = The Learning rate. where $1 - \alpha$ is the progress ratio.

Assume for example that the cost of developing the first product is 100 and the learning rate ($\alpha$) is 90%, then according to the above formula the cost of developing the second product is 90, the cost of developing the third product is 84.6, the fourth product is 81, and so on. It appears that for every doubling of the input the cost for developing the product reduces with 10%, which defines the so-called *progress ratio* ($1 - \alpha$). The experience curve effect shows that experience has a direct impact on the productivity of a task. Different learning rates will have a different impact on the productivity. Fig. 1 shows, for example, the experience curve for three different learning rates (90%, 80%, and 70%). The left figure shows the linear version, while the right figure shows the logarithm version of the three curves, resulting in straight lines. Note that for higher learning rates (90%) the direct cost per unit decreases much faster than for lower learning rates (70%). In this case, the higher the percentage of the learning rate $\alpha$, the lower the progress rate ($1 - \alpha$), and as such the lower the productivity.

Depending on the characteristics of the required tasks different industrial domains appear to have their own specific learning rate, ranging from 75% to 95% [11]. Example domains together with their derived learning rates are shown in Table 1. In general, the nature of the tasks defines the experience curve. Hereby, it appears that the learning curve rate is lower (thus progress rate is higher) for domains in which tasks requires more manual intervention.

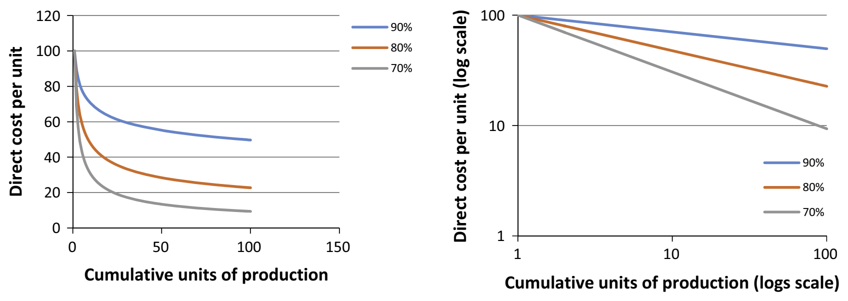

**Fig. 1.** Experience curve shown in linear–linear and log–log forms.

**Table 1**
Different domains with their learning rates.

| Domain | Learning rate (%) |
|---|---|
| Aerospace | 85 |
| Shipbuilding | 80–85 |
| Complex machine tools for new models | 75–85 |
| Repetitive electronics manufacturing | 90–95 |
| Repetitive machining or punch-press operations | 90–95 |
| Repetitive electrical operations | 75–85 |
| Repetitive welding operations | 90 |
| Raw materials | 93–96 |
| Purchased parts | 85–88 |

**Table 2**
Publication sources searched.

| Source | Number of included studies after applying search query | Number of included studies after exclusion criterion |
|---|---|---|
| IEEE Xplore | 65 | 2 |
| ACM Digital Library | 19 | 1 |
| Wiley Interscience | 31 | 0 |
| Science Direct | 13 | 1 |
| ISI Web of Knowledge | 58 | 4 |
| Other channels | 974 | 7 |
| Total | 1160 | 15 |

## 3. Systematic literature review on experience curve effect

### 3.1. SLR protocol

The experience curve has a direct impact on the productivity in different domains. An important question is whether this also holds for the software engineering domain. To investigate the experience curve effect on software development we have conducted a systematic literature review using the guidelines as described by Kitchenham and Charters [12]. In particular we were interested in the answers to the following research questions:

- *RQ1: For which goals has the experience curve effect been studied in software engineering?*
- *RQ2: In which domain contexts has the experience curve model been applied in software engineering?*
- *RQ3: What is the reported evidence in the application of the experience curve models in software engineering?*
- *RQ4: In which software lifecycle activities has the experience curve model been applied?*
- *RQ5: What are the identified findings with respect to experience curve effect in software engineering?*

Our search scope included all the papers that were published before July 2014. We searched for full papers in selected venues that publish high quality papers. We used the following search databases: IEEE Xplore, ACM Digital Library, Wiley Inter Science Journal Finder, ScienceDirect, ISI Web of Knowledge, and other channels including Microsoft Academic Search and manual search channels. These venues are listed in Table 2. Our targeted search items were journal papers, conference papers, and workshop papers.

To search the selected databases we used both manual and automatic search strategies. Automatic search is realized through entering search strings on the search engines of the electronic data source. Manual search is realized through manually browsing the conferences, journals or other important sources and checking the references of selected papers. The manual searches appeared to be quite useful since we retrieved some good-quality articles that an automatic search could not reveal.

The adopted search string was as follows:

*("experience curve" OR "learning curve") AND ("software development" OR "software engineering" OR "software process" OR "software project")*

The result of the overall search process after applying the search queries and the manual search is shown in the second column of Table 2. As it can be seen from the table we could identify 1160 papers at this stage of the search process. We have also run a similar query to check if the effect of experience curve is discussed in the SPLE context. We ran a similar analysis with ("software product line" OR "software product family") AND ("experience curve" OR "learning curve") keywords. After the initial set of exclusion, we were unable to find any papers that discuss this issue.

In accordance with the SLR guidelines [12] we further applied the following exclusion criteria on the large number of papers in the first stage:

- Abstract or title does not primarily discuss the effect of "experience curve" on software development productivity
- Not a primary study
- Repeated in an already mined source
- Most of the content is repeated in a similar paper (Extended version is chosen over the shorter one)

The exclusion criteria were checked by a manual analysis by both the authors. To the best of our knowledge, no other secondary study has been published related to experience curves in software development. After applying the exclusion criteria to the identified 1160 papers we selected 15 papers for a detailed analysis.

### 3.2. Data extraction results

For data extraction and synthesis process as required by the systematic review protocol we thoroughly studied the primary studies in detail to answer our five defined research questions. The identified primary studies and the result of the data extraction and synthesis process are summarized in Table 3. The answers to the research questions RQ1 to RQ4 are directly shown in the table. The specific answers to RQ5 are described in the following paragraphs in which we provide a short summary of each identified primary study with the basic conclusions:

### 3.2.1. A learning curve primer for software engineers [13]

The basic aim of this paper from 1996 is to raise the level of understanding of learning curves within the Software Engineering community. The paper is primarily written for managers and developers who want to better understand learning curves. The author first defines learning curves, provides the historical background and argues that learning curves also apply to software engineering. The paper further elaborates on the challenge of both stabilizing and improving a software development process. The paper argues that learning curves denote the relationship between stability and improvement, and show the implications of learning curves on staffing a project. Based on these findings the paper shows that Fred Brooks's observations about man-months can be explained in terms of learning curves, and that it is needed to keep teams together on

**Table 3**
The identified primary studies and the result of the data extraction and synthesis process.

| Study | RQ1<br>Goal | RQ2<br>Domain context | RQ3<br>Validation approach | RQ4<br>Life cycle activity |
|---|---|---|---|---|
| A learning curve primer for software engineers [13] | General understanding and introduction of the effects of learning curve in software development process | NA | General discussion<br>No validation | Development |
| Human oriented improvement in the software process [14] | Investigate the impact of second-order learning (induced learning) on progress in software development | Small-sized student projects | Experiment using 12 student software developers, who completed one small-sized project | Development |
| Learning and forgetting curves in software development [15] | Investigate the impact of learning curve and forgetting curve for different types of knowledge including domain, technology and methodology | IT Service | Retrospective Analysis from sample of 556 projects from 2005–2007 | Development |
| Learning from experience in software development – multi-level analysis [16] | Investigate impact of the accumulating specialized experience in a system, diversified experience in related and unrelated systems, and experience from working with others on productivity | Telecommunications industry | Retrospective analysis of covering more than 14 years of systems development work on a major telecommunications product | Maintenance |
| A matter of balance: specialization, task variety, and individual learning in a software maintenance environment [17] | Investigate (1) how experience enhances individual learning and impacts productivity? (2) the role of variety in individual learning (3) the impact of the composition of an individual's overall portfolio of experience on learning and productivity | Offshore software support services operation | Retrospective analysis covering a data set covering 88 individuals who worked on 5711 maintenance tasks over a period of six years | Maintenance |
| A learning curve based simulation based model for software development [18] | Investigate the impact of learning curve on software development to support estimating and/or predicting the development time and the total work effort in project planning | NA | Simulation using four different scenarios including parameter values for number of developers, learning curve rates and activity property | Development |
| An empirical study of ICASE learning curves and probability bounds for software development effort [19] | Investigate the existence of learning curves in software development in general, and impact of programmers experience in an integrated Computer-aided software engineering (ICASE) tool on the software development effort | Software projects from Texas Instruments and Electronic Data Systems | Retrospective analysis of data set covering forty projects obtained from two major companies in the North Eastern USA | Development |
| Optimal allocation of testing effort during testing and debugging phases a control theoretic approach [20] | Investigate optimal allocation of testing effort during testing and debugging phases considering learning curve effect | NA | Analytical evaluation | Testing |
| Virtual organizational learning in open source software development projects [21] | Investigate whether learning effect is universally present in Open Source Software (OSS) projects, and identify the factors that affect the learning process | Open Source Software Projects | Retrospective analysis of number and percentage of resolved bugs and bug resolution time of 118 SourceForge.net OSS projects | Maintenance |
| A quantitative learning model for software test process [22] | Define a novel quantitative learning model for software test processes | IT Software | Analysis of a tool that transforms Cobol Code to SAP/R3 and interviews with project manager | Testing |
| Organizational learning in open-source software projects: an analysis of debugging data [23] | Investigating the learning curve effect in open source software projects | Open Source Software Projects | Retrospective analysis by analysis of debugging data of Apache and Mozilla projects | Maintenance |
| A learning curve explanatory theory for team learning valuation [24] | Introducing a theory for team learning valuation | Mixed | Retrospective analysis of ISBG database software projects | Development |
| A comparative analysis of learning curves implications for new technology implementation management [25] | Examining the impact of learning curve theory on ERP implementation planning and management | ERP | Empirical analysis of four ERP implementation projects in three companies | Development |
| Quantitative control of process changes through modeling, simulation and benchmarking [26] | Provide a comparison between productivity learning curve model derived by Motorola and COCOMO II | NA | Simulation of the models with different parameters | Development |
| People applications in software process modeling [28] | Explores the role of people in process modeling including the learning curve theory aspect | NA | No Validation | Development |

long-term projects because of the learning curve effect. In the final section of the paper different equations for learning curves are provided that can be used to model a process and measure the impact of learning curves on software engineering projects.

### *3.2.2. Human oriented improvement in the software process [14]*

This paper discusses the so-called *progress functions* that differ from *learning curves* because the former also incorporates a *second-order* learning mechanism. Whereas first-order learning (also referred to as *autonomous* learning) is the improvement due to the experience, which a person gains by repeatedly doing the same task, second-order learning (also known as *induced* learning) is due to the technologies injected by the organization. The paper makes an explicit distinction between learning curve and progress function and aims to analyze these separately. For this, the paper describes the setup and results of laboratory experiment for analyzing the impact of progress due to second-order learning. The described experiment involved a sample of 12 student software developers, who completed one small-sized project every week for ten weeks. A within-subject, repeated-measure, time-series quasi-experimental design was used as the research method. The statistical tests showed that on average, progress takes place at a rate of about 20%, with technology injection this amounts to 13% improvement over autonomous learning alone. The authors claim that such a distinction is useful for making decisions regarding initiating formal training programs and making engineering technology changes.

### *3.2.3. Learning and forgetting curves in software development [15]*

In this study, the authors aim to explore what type of knowledge, among *domain*, *technology*, and *methodology* knowledge, is most influential to the performance of software development. To answer this question the paper presents the results of an empirical study of the impact of learning and forgetting curves in software development using archival data from sample of 556 software development projects in an IT service company. This sample of projects involved 3341 unique employees, 6675 employee-project assignment records, and 206,173 employee project-technology records. The basic findings of the paper are that prior experiences with the same methodology or technology have a stronger impact on software project performance than those in the same application domain. Furthermore, the results of the study show that methodology knowledge is more easily forgotten than domain or technology knowledge. The authors argue that these insights have implications for the development of knowledge and skills as well as other organizational issues in software development such as project team staffing and career development.

### *3.2.4. Learning from experience in software development – multi-level analysis [16]*

This study discusses the impact of the learning curve on individuals, groups, and organizational units learn from experience in software development and whether this learning improves productivity. The study evaluates the relative productivity impacts from accumulating specialized experience in a system, diversified experience in related and unrelated systems, and experience from working with others on modification requests (MRs) in a telecommunications firm. The study includes the analysis of data archives covering more than 14 years of systems development work on a major telecommunications product dating from the beginning of its development process. The findings of the study show that the relative importance of the different types of experience differs across levels of analysis. According to the study specialized experience has the greatest impact on productivity for MRs completed by individual developers, whereas diverse experience in related systems plays a larger role in improving productivity for MRs and system releases completed by groups and organizational units. Diverse experience in unrelated systems has the least influence on productivity at all three levels of analysis. The paper concludes that learning curves have an impact in software development and, in particular provides insights in when specialized or diverse experience may be more valuable.

### *3.2.5. A matter of balance: Specialization, task variety, and individual learning in a software maintenance environment [17]*

The primary study aims to provide answer to the following questions. (1) How does the pattern of past experience enhance individual learning and productivity? (2) What is the role of variety in individual learning? (3) How does the composition of an individual's overall portfolio of experience affect learning and productivity? To answer these questions the authors have analyzed 88 individuals who worked on 5711 maintenance tasks in an offshore software support services operation. The study concludes that specialization enhances productivity in general, and a proper balance between specialization and exposure the variety is needed to achieve the highest productivity. A further conclusion of the study reveals that the degree of variety experience lost has a greater impact on productivity than the degree of specialized experience that is lost.

### *3.2.6. A learning curve based simulation model for software development [18]*

In this article, the authors propose a simulation model for software development that takes into account the developer's learning curve. As such it can be used to compute a developer's productivity and the quantity of gain to the developer's knowledge in executing an activity. Based on this simulation model, a project-planning prototype has been implemented. The proposed model and the prototype have been applied to four typical scenarios in a case study. The presented prototype can help managers to decide whether the new software development environment should be used in the project or not.

### *3.2.7. An empirical study of ICASE learning curves and probability bounds for software eevelopment effort [19]*

In this article, the authors investigate the existence of learning curves in software development. The authors illustrate this by examining the relationship between a programmer experience in an integrated Computer-aided software engineering (ICASE) tool and the software development effort. They have validated their approach on actual software development effort in forty projects obtained from two major companies in North Eastern USA.

### *3.2.8. Optimal allocation of testing effort during testing and debugging phases a control theoretic approach [20]*

The main goal of this paper is to investigate an optimal resource allocation plan to minimize the cost of software during the testing and operational phase. During the analysis, it has been observed that due to the experience curve phenomenon, the effort required to fix an error keeps on decreasing with time. At the same time, testing effort keeps on increasing as in the later stages of a planning period it becomes hard to detect faults. Based on this fact they propose an alternate rationale for optimal allocation of testing resources using learning curve phenomenon under dynamic environment.

### *3.2.9. Virtual organizational learning in open source software development projects [21]*

The study aims to provide answers to the following two research questions: (1) Are learning effects universally present in Open Source Software (OSS) projects? (2) What are the factors that affect the learning process? For this, the authors used the number and percentage of resolved bugs and bug resolution time to measure learning effects. Further they also looked at how different project types, number of developers (project team size) and their experience, and the intensity of assigned bugs affected the learning rates. The data for this study were obtained from the SourceForge.net database. The findings of the paper show that both adaptive and organizational learning were observed in the projects. Further, the results also showed that OSS development project performance was influenced by the number of developers on the team, the amount of experience that they possessed, project category, and percentage of bugs assigned to a person. Finally, the results indicated that though smaller teams learned faster, they suffered from greater variability in efficiency.

#### 3.2.10. A quantitative learning model for software test process [22]

The authors investigate the learning curve effects on software test processes and compare existing learning models. An existing formal software test process model is modified to include the effects of learning. The updated software test process model with an integrated learning curve is applied to several industrial software test projects for validation. The authors conclude that the larger the learning phase, the better the predictions of improved model when compared to the previous model.

#### 3.2.11. Organizational learning in open-source software projects: an analysis of debugging data [23]

This paper analyzes the organizational learning effect in open-source programming projects. For this, the study uses the data from Apache and Mozilla projects to analyze bug cycle times. The authors conclude that despite of the similarity among the projects there are significant qualitative and quantitative differences in their learning processes. The authors observed a higher impact of the learning curve for a mature project compared to an emerging project. This observation implies that the success of any given learning approach is likely to be situational, and should be examined in the context of the given organization.

#### 3.2.12. A learning curve explanatory theory for team learning valuation [24]

This paper examines how the learning curve effect can be used to derive monetary information for team learning observed within knowledge-intensive production environments. For this, software development is selected as an identical example of a team-based, knowledge-intensive production environment. The interaction of learning rate of the developer teams and the improvements on the productivity is modeled as a Lotka–Volterra predator–prey interacting populations system establishing a causal relationship between the human capital (HC) of organizational teams and the observed learning curve effects on their performance. The theoretical justification is supported by empirical evidence based on the ISBSG database.

#### 3.2.13. A comparative analysis of learning curves implications for new technology implementation management [25]

This paper examines how the learning curve theory would affect better planning and management of ERP implementation projects. In the literature, there have been several studies that discuss the effect of team training/learning on software implementation projects' schedule and cost. Based on this the authors explore the learning curve methods for adopting in ERP project management. Using empirical data from 4 ERP projects, the authors provide a comparison between general logistic and S-curves in learning curve theory. The authors conclude that a logistic curve is a good approximation for the majority of the ERP implementations.

#### 3.2.14. Quantitative control of process changes through modeling, simulation and benchmarking [26]

In this paper, the authors have evaluated the productivity learning curve as defined by Motorola [27] and compared it with the COCOMO II model cost drivers. The selected COCOMO II cost drivers APEX, LTEX, PLEX are calibrated to represent learning curve effects. The paper lists the advantages and disadvantages of using both models when representing learning curve effects for measuring software development productivity.

#### 3.2.15. People applications in software process modeling [28]

This paper discusses the importance of people factors and human relations above technical aspects for improving productivity in software development. A list of people factors are provided including motivation, exhaustion, experience and learning curves, training, hiring and retention, communication, stakeholder collaboration and workforce dynamics.

### 3.3. Threats to validity

The findings from the SLR could have suffered from several validity threats. Below we describe the potential threats and briefly discuss the mitigation strategy for each threat.

*Construct validity* refers to the degree to which the SLR measures what it aims to be measuring. One possible threat to construct validity is exclusion of relevant studies. In order to minimize this threat, we applied detailed guidelines of systematic literature review protocol and defined a rigorous search strategy. In our SLR we used the important search databases: IEEE Xplore, ACM Digital Library, Wiley Inter Science Journal Finder, ScienceDirect, ISI Web of Knowledge, and Google Search. We also searched for company journals, grey literature, conference proceedings and the internet, which led us to new papers that we could not identify in our regular search. We performed the inclusion/exclusion procedures on a well-established screening of primary studies. In our search we have only considered the title and abstract because in the other case the false positives are dramatically large. In particular the term learning curve is a widely used daily term that is not directly related to the scientific concept that we refer to. For reducing the selection bias for selecting the primary studies, the evaluation and the selection of the primary studies were performed separately by two researchers. Each researcher recorded also the reasons of acceptance or rejection for all the considered studies. Despite of the careful and in-depth search we realize that we might have still missed some studies. Nevertheless, we believe that we have captured the important studies that discuss learning curve and that are of value to discuss the impact of the learning curve on the ROI in SPLE.

*Internal validity* threat includes the possibility of deriving an invalid causal relationship based on the findings. In the context of the present study, the SLR was essentially exploratory, with the aim to identify answers to the corresponding research questions. Hence, it was sufficient to ensure that the selected primary studies were relevant in the literature.

*Conclusion validity* (reliability) is the degree to which conclusions about the relationship among variables based on the data are correct or reasonable. This threat is mitigated by adopting a clear SLR protocol including well-defined steps and the involvement of two researchers. The outcome of the SLR is quite broad and if the study would be replicated by different set of researchers it is possible that the final set of selected primary studies could be slightly different, but the general findings would be quite similar. As such, we believe that the conclusion validity of the SLR is high given the use of a very systematic procedure and the involvement and discussion among the two researchers.

*External validity* refers to the extent to which the results of the SLR can be generalized outside the scope of the study. Within the context of our study we can relate this to the degree to which the primary studies and the extracted data elements are representative of the overall goal of the review. This risk is largely mitigated by the detailed and careful review protocol that we have discussed above.

## 4. Software product line engineering cost models

The overall conclusion of the systematic literature review is that the experience curve has a direct impact in software development. As we have discussed in the previous section this conclusion has been drawn from research in various contexts in software development. Unfortunately, the experience curve effect in SPLE in particular has not been directly addressed before. For transitioning to SPLE it is important to define the proper cost models for providing the cost-benefit analysis of adopting an SPLE approach. In current transition strategies that adopt different cost models the experience curve effect has not been taken into account. Very often, this is not realistic in practice and can lead to a wrong decision making in the SPLE adoption process. In the following we first take a closer look at the cost models using the experience curve effect for both single development and SPLE, and then extend these cost models to provide a more precise analysis to support the decision making process.

The following represents the general formula for calculating the cost of adopting SPLE as discussed in the Structured Intuitive Model for Product Line Economics (SIMPLE) [8,29]:

$$C = C_{Org} + C_{Cab} + \sum_{i=1}^{n} (C_{Unique}(p_i) + C_{Reuse}(p_i)) \tag{2}$$

In the formula, $C$ represents the overall cost and consists of the cost of adapting the software reuse approach for the organization ($C_{Org}$), the cost to define the asset base ($C_{Cab}$), the sum of the cost for developing unique portion of the products ($C_{Unique}$) and the sum of the cost of reusing core assets ($C_{Reuse}$) for a software product line of $n$ products denoted by $p_i$.

For calculating the development cost in single system development only the cost for developing unique products is needed, ($C_{Org}$), ($C_{Cab}$), and ($C_{Reuse}$) can be omitted. As such, the cost model for this situation is shown in Eq. (3).

$$C = \sum_{i=1}^{n} (C_{Unique}(p_i)) \tag{3}$$

Based on the general cost models, several other cost models for different scenarios have been defined [7,8]. An important point in the process of adopting an SPLE approach is the break-even point after which the organization will start to get the ROI for the upfront investment. Considering either the cumulative cost for developing products or the time needed to deliver the products to the market the ROI will be expressed differently. This is shown in Fig. 2.

The left figure shows the ROI for the cost of developing the products, while the right figure shows the ROI with respect to time. Typically finding the break-even point is critical for an organization, because this will usually guide the decision for adopting SPLE or not. The exact location of the break-even point depends on various characteristics such as the organization and market characteristics, the range and kind of products, and the selected transition strategies. However, in the SPLE literature the break-even point is generally reported between 2 and 3 products [2–4,30]. The ROI of SPLE can be calculated by dividing the Cost Savings with Cost of investment. Cost savings can be calculated by subtracting the cost of SPLE approach (Eq. (2)) from the cost of single development (Eq. (3)). The cost of investment is ($C_{Org} + C_{Cab}$). As such, the formula for ROI can be expressed as follows:

$$\frac{\sum_{i=1}^{n} C_{Unique}(p_i) - (C_{Org} + C_{Cab} + \sum_{i=1}^{n}(C_{Unique}(p_i) + C_{Reuse}(p_i)))}{C_{Org} + C_{Cab}} \tag{4}$$

Eqs. (2)–(4) have been derived from the SPLE literature. As it can be observed none of these explicitly consider the experience curve effect. We elaborate on this in the next section.

## 5. Integrating experience curve effect in cost models

To analyze the impact of experience curve on SPLE we will consider both the equations for the experience curve (Eq. (1)) and the equations for the cost models of SPLE as defined in Eqs. (2)–(4). For this we will adopt the following scenario that is derived from the scenario template as defined in [8]:

*"An organization of size s, plans to bring k products to the marketplace but hasn't begun developing them yet. It wants to explore building them as a software product line from a common set of core assets."*

For this scenario, we will first define the cost models with the experience curve effect, and then use these cost models to compare the cost of developing a set of products both in SPLE approach, and single-system development approach.

### 5.1. Cost models with experience curve effect

In the cost model for SPLE (Eq. (2)) we can identify several cost elements that can be influenced by the experience curve. Since $C_{Org}$ and $C_{Cab}$ are non-repeating activities we assume that the experience curve has a negligible or no effect for these elements. The elements $C_{Unique}$ and $C_{Reuse}$, however, are repeating activities and will be typically influenced by the experience curve effect. As such, we can now adapt the formula Eq. (2) by integrating the experience curve effect as defined in Eq. (1). The result is as follows:

$$C = C_{Org} + C_{Cab} + \sum_{i=1}^{n} (C_{Unique}(p_1) i^{e1} + C_{Reuse}(p_1) i^{e2}) \tag{5}$$

where

$C_{(unique)}(p_1)$: is the cost of building the unique parts of the first product
$C_{(reuse)}(p_1)$: is the cost of building the common parts of the first product using core asset base
$n$: is the number of products in the product line
$e1$: is the learning index for developing unique portions of different product in the same product family
$e2$: is the learning index for developing reused portions of the different products in the same product family

In fact Eq. (5) can now be considered as a more general equation for calculating the product development costs. In case the experience curve effect is neglected it will represent the conventional equation Eq. (2). The cost model of Eq. (5) is defined for the given scenario in which product line is set up from the beginning (Big Bang strategy). However, other scenarios as defined in [8] could be adapted similarly.

We can also derive the cost model with the experience curve effect for single system development. For this $C_{Org}$, $C_{Cab}$ and $C_{Reuse}$ will be zero, and likewise we will get the following formula:

$$C = \sum_{i=1}^{n} (C_{Unique}(p_1) i^{e}) \tag{6}$$

In this formula, $C$ is the accumulated cost of building n products by considering the experience curve effect, where $C_{Unique(p1)}$ is the cost

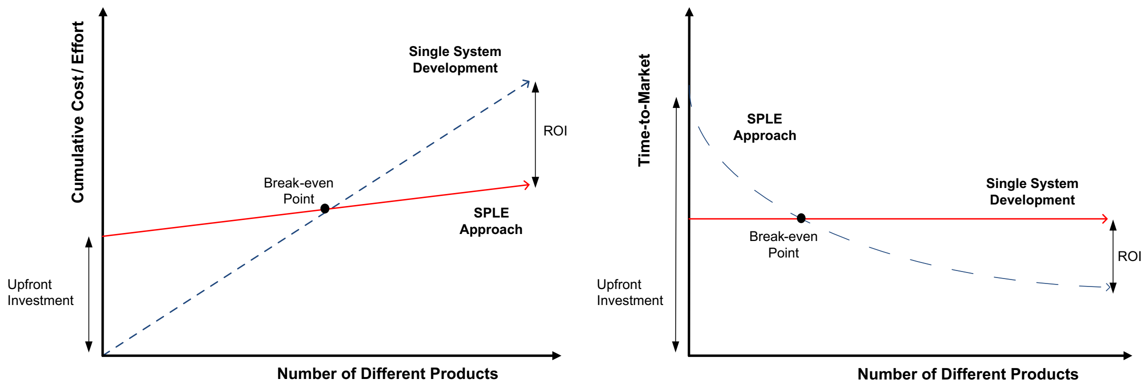

**Fig. 2.** Typical ROI figure for software product lines.

of building the first product, $n$ is the number of products, and $e$ is again the learning index for developing similar software. The ROI of SPLE with the experience curve effect can now be defined as follows:

$$\frac{\sum_{i=1}^{n}(C_{Unique}(p_1)i^{e}) - [C_{Org} + C_{Cab} + \sum_{i=1}^{n}(C_{Unique}(p_1)i^{e1} + C_{Reuse}(p_1)i^{e2})]}{C_{Org} + C_{Cab}} \tag{7}$$

### 5.2. Analysis with refined cost models

Obviously, the refined cost models will have an impact on the break-even point and the ROI. For calculating the cost formulas of the given scenario in the introduction of this section, different values can be provided for the parameters $s$ (organization size) and $k$ (number of products). In the following we provide an example calculation for organization of size 30 ($s$), which wishes to develop 8 products ($k$). For the parameter values of the cost model we adopt the constants that are given in [8], and which are shown in Table 4. Here, $C_{Prod}$ represents the cost of building a single product, $C_{Org}$ represents the cost of changing the organization to adopt SPLE, $C_{Scop}$ represents the cost for carrying out a scoping exercise to determine the commonality and variability analysis for determining the core asset base. The remaining terms are derived from these three constants. In the cost models of Eqs. (5) and (6) we have also the parameter values $e$, $e1$ and $e2$ related to the experience curve. For defining the values for these experience curve parameters we use the constant of 0.8 in alignment with the literature and previous work on the effects of experience curve in software development.

Once all the parameter values are defined we can calculate the cost and define the ROI for the given scenario. Table 5 lists the result of the calculation for the cost of producing the $i$th product (in man-months). We have calculated the values for the cost models as defined in Eqs. (2), (3), (5), and (6).

In Fig. 3 we can observe that the experience curve effect reduces the cost of development for both single development and SPLE. From this figure we can also observe that the experience curve effect leads to a reduction of the cost of development. Hereby, SPLE with experience curve effect is the most optimal for reuse and cost reduction. In addition, we can also state that the experience curve effect will slowly diminish in due time.

According to Eq. (7), we compared the cost models with and without integrating the effects of experience curve, the results are shown in Table 6. For SPLE it appears that the break-even point is achieved later. In the cost model without experience curve effect the ROI will be achieved with the second product. For the refined cost model with experience curve effect the ROI is achieved together with the third product. The experience curve effect seems to have a clearly visible effect on the ROI.

The above analysis provides the results of the calculation for a given set of values. The analysis can be used to simulate various different cases to estimate the cost of SPLE adoption. Given an organization the parameter values for the organization sizes, the number of products ($k$), the different cost values for $C_{Org}$ and $C_{Cab}$, and even the learning curve rates can be adopted to support the cost estimation of the SPLE adoption.

## 6. Application within industrial project

To illustrate the application of the refined cost models and the related analysis we describe the results of a real industrial case study. We also discuss the implications of the application of the refined analysis on the decision making of the adoption of SPLE. Finally, we describe the threats to validity of the application of the approach to the industrial case.

### 6.1. Industrial case

We have applied the previous analysis at Havelsan, which is a leading Turkish software and systems company having business presence in IT sector. The company operates in three main business areas: command and control (C2), simulation and training systems, and e-government systems. In the command and control division three products were developed for different customers and the fourth product is under development. The development language and adopted tools were the same across all projects. The development team was largely the same during all three projects. So far, the products have been developed using a non-SPLE approach. From a practical perspective the development of the second product was realized in a shorter time of period than the first product. On its turn the third product was realized in a shorter time than the second product, indicating the impact of the experience curve effect. In fact, the developed and planned products share lots of commonality but an SPLE approach was not adopted. Moreover, due to the stringent requirements for time-to-market and cost of development, the management decided to analyze whether

**Table 4**
Variable definition and values [8].

| Term | How To compute | Value |
|---|---|---|
| $C_{Prod}$ | ~12 person-years(PY) per product | ~12 PY |
| $C_{Org}$ | Organization Size * 1 person-month(PM) | ~2.4 PY |
| $C_{Reuse}$ | 70% * 10% * $C_{Prod}$ | ~0.84 PY |
| $C_{Unique}$ | 30% * $C_{Prod}$ | ~3.6 PY |
| $C_{Cab}$ | 150% * 70% * $C_{Prod}$ + $C_{Scop}$ | ~13 PY |

**Table 5**
Cost comparison of different cost models for given scenario.

| Number of products | SPLE dev. (Eq. (2)) | Single dev. (Eq. (3)) | SPLE development with exp. curve (Eq. (5)) | Single development with exp. curve (Eq. (6)) |
|---|---|---|---|---|
| 0 | 186.2 | 0 | 186.20 | 0.00 |
| 1 | 239.48 | 144 | 239.48 | 144.00 |
| 2 | 292.76 | 288 | 286.42 | 259.20 |
| 3 | 346.04 | 432 | 330.06 | 360.30 |
| 4 | 399.32 | 576 | 371.50 | 452.46 |
| 5 | 452.6 | 720 | 411.33 | 538.23 |
| 6 | 505.88 | 864 | 449.89 | 619.12 |
| 7 | 559.16 | 1008 | 487.42 | 696.08 |
| 8 | 612.44 | 1152 | 524.07 | 769.81 |

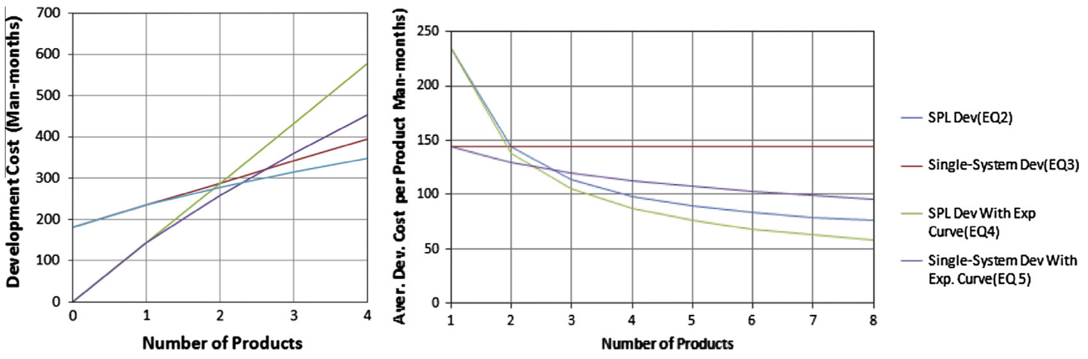

**Fig. 3.** Cost comparison of single system development and product line engineering, with and without the experience curve effect.

**Table 6**
ROI comparison for both cost models.

| Number of products | ROI without experience curve effect (%) (Eq. (4)) | ROI with experience curve effect (%) (Eq. (7)) |
|---|---|---|
| 0 | −100.00 | −100.00 |
| 1 | −51.28 | −51.28 |
| 2 | −2.56 | −14.62 |
| 3 | 46.17 | 16.24 |
| 4 | 94.89 | 43.48 |
| 5 | 143.61 | 68.16 |
| 6 | 192.33 | 90.88 |
| 7 | 241.05 | 112.07 |
| 8 | 289.77 | 131.98 |

**Table 7**
Analysis of cost with and without considering experience curve effect in the context of Havelsan.

| Number of products | Without experience curve effect (estimated man-month) | With experience curve effect (estimated man-month) | Measured actual value (normalized man-month) |
|---|---|---|---|
| 1 | – | – | 75 |
| 2 | 75 | 61.54 | 63.49 |
| 3 | 75 | 54.81 | 56.25 |
| 4 | 75 | 50.49 | NA |
| 5 | 75 | 47.38 | NA |
| 6 | 75 | 44.97 | NA |
| 7 | 75 | 43.04 | NA |
| 8 | 75 | 41.43 | NA |

adopting an SPLE approach for the subsequent products would pay off. For this we applied the analysis as described in the previous sections.

The organization is a CMMI Level 3 company and likewise we did not have serious problems to retrieve the required historical data (man-month cost) that we collected from the company's time-tracking system (SAP). For the analysis first we calculated the learning rate of the corresponding team as 0.82, which was based on data from earlier projects. Next we focused on providing the analysis of the system with and without using the experience curve effect. The results of the analysis are shown in Table 7. The first column represents the products that were developed or are planned to be developed. The second column of the table shows the estimated cost values in case experience curve effect is not taken into account. The third and fourth column of the table show respectively the estimated and measured cost values based on the experience curve effect. We could derive the actual measured values for the first three products since these were already developed. The cost of the first product was 75 man-months (mm); this was directly measured using the time tracking system. Since single system development is used and experience curve effect is not taken into account, the cost value for each product is assumed to be constant and this explains the values for the second column. In the third column we can see that the calculated cost values are increasingly lower than the cost of the first product (75 mm) and this shows the impact of the integration of the experience curve effect in the analysis. For the management it was important to know whether these calculated values were accurate and reflected the reality. The actual values of the fourth column seemed to be close to the calculated values and the error rate of the calculations was less than 5%. This error rate may be due to different factors, such as development team changes, and external factors affecting performance. Based on this input the calculated values showed to be reliable.

Using this information we have applied Eq. (7) to find the ROI, which is shown in Table 8. We applied Eq. (7) with the following parameters: Organization size: 7; Commonality between products: %67.5; $e$, $e1$ and $e2$ as 0.8205.

According to Table 8, with the original formula without the experience curve effect, it is economically feasible to apply SPLE development in case 2 products are developed. Using the formula with the experience curve effect the ROI was achieved with the third product. This confirms the case for applying SPLE. In this case, it would have been indeed more beneficial for the company if an SPLE approach would have been adopted from the beginning on.

**Table 8**
ROI for the C2 products.

| Number of products | ROI without experience curve effect (%) | ROI with experience curve effect (%) |
|---|---|---|
| 0 | −100.00 | −100.00 |
| 1 | −47.02 | −47.02 |
| 2 | 5.96 | −3.55 |
| 3 | 58.94 | 35.17 |
| 4 | 111.92 | 70.84 |
| 5 | 164.90 | 104.30 |
| 6 | 217.88 | 136.07 |
| 7 | 270.86 | 166.48 |
| 8 | 323.84 | 195.74 |

On the other hand according to the new formula with the experience curve effect, SPLE provides a positive ROI after the third system. This slightly favors the adoption of single development due to the impact of the experience curve effect.

### 6.2. Impact on decision making for adopting SPLE

The adapted cost formulas with the experience curve effect provided new insight to the management and it was decided to integrate this analysis in the further decision making process. The theoretical formulas that we have defined in the different equations were not a problem because we implemented all these equations as a set of Excel sheets. The Excel sheets now forms an important instrument for the management to do a cost benefit (ROI) analysis in the Havelsan projects. The usage of the Excel sheet is straightforward and requires only the input of the basic parameters. The charts as shown in Fig. 3 are automatically generated.

Initially, it was believed that the experience curve effect would form an important argument for continuing with single system development and adopting SPLE was actually not seriously considered. However, our analysis showed that adopting SPLE would still pay off in the end, even with considering the experience curve effect. This was an eye-opener for the management, and the decision to transition to SPLE for potential projects was now much easier. As stated before, this was also supported by the provided other benefits of SPLE including increased quality and reduced maintenance costs.

### 6.3. Threats to validity

We have carefully applied the refined cost models to the industrial case study of Havelsan and as we have discussed in the previous subsection we could directly observe the impact on the decision making. Like any empirical study we can also identify several threats to the validity of this analysis which we list below:

*Internal Validity* concerns the ability to draw conclusions between the independent and dependent variables in the study. Our basic independent variable in the study is the experience curve effect, while the dependent variable is the development cost. Other variables that could have an impact on the development costs in the industrial case are the team, the work environment, the adopted tools and the products that were developed. We have ensured that all these parameters remained the same in the projects, and the developed products were similar products of the company. Since the product sizes were not equal in most cases, we also applied normalization approach to be able to compare the costs of between different products. For this, we both checked SLOC size and requirements size of the products.

*Construct Validity* concerns the appropriateness of the measures for capturing the dependent variables. In our analysis we measured the size of the products and the man-months to measure the development costs. We measured the size data by looking at the requirements documents and SLOC values of the produced software in the company. We counted the SLOC from the version control repositories using automated scripts. The requirements documents are documented based on the CMMI Level 3 requirements and hence are substantially reliable. The development team's effort data for the projects was collected from the company's time-tracking system, which accurately represents the costing data. We could not measure the values for SPLE experience curves since these were not available before the SPLE adoption process. Hence we used the calculations, which were based on assumptions of earlier paper by Boeckle et al. [8].

*External Validity* concerns the ability to generalize the results of the study. The formulas for the cost models for SPLE and experience curve effect have been derived from the literature. In the systematic literature review as described in Section 3 we have shown that both types of cost models have been analyzed thoroughly and validated earlier on. Our cost model that integrates both formulas applies the experience curve effect for repeating tasks which were identified as the cost of developing individual products and reuse of products. This application is in alignment with the usage of the experience curve effect in other contexts. Our findings are based on the analysis of three real industrial products, which were developed over two years. Hence the validation of the study is not based on a toy project or simple academic case studies. We applied a retrospective study for the analysis and the team members were not aware that the project would be analyzed later on for our purposes.

## 7. Discussion

The systematic literature review has shown that the experience curve effect has also an impact in software development. Using the SPLE cost models we have proposed and demonstrated the usage of the extended cost models with the experience curve effect. Based on our analysis and industrial experiences with the approach we can derive several general practical implications for the decision making process in the SPLE adoption process.

- *Due to similar repeating tasks in SPLE the experience curve effect will be observable*

The experience curve is valid under the assumption that the same knowledge is applied repeatedly. In case the knowledge is outdated or novel knowledge is applied, a new experience curve is required. Typical experience curve examples are about producing the same unit. In SPLE the second "similar" project/product that is assumed to be in the same product family will not be identical, however it can be argued that the second project/product is very similar due to the common platform (same set of tools, processes, technology, etc.) Here it is critical to assess the similarity between the products. Schilling et al. [31] suggest that people working on different but similar types of problems over time, on the average learned at a significantly faster rate than people work at specialized tasks, or unrelated tasks. This means fine-grained variation between products in a product line even has a positive impact on the effect of learning. In the context of SPLE we can assume that similar tasks are defined and as such the assumptions on experience curve effect will have an impact here as well.

- *The experience curve has a clear impact on the ROI*

Our study and the scenarios that we have adopted show that the experience curve has a clear impact on the cost models and likewise the ROI. From a theoretical perspective we relied on the conclusions of the systematic review and adopted estimations of the learning rate as described in the primary studies. The application of the experience curve to cost models as adopted in the industrial setting also showed the impact of the experience curve, and led to decreasing time and cost to develop products. Although this observation of experience curve seems to be intuitive it can have an impact on the decision making process in case one needs to decide whether to keep the single system development approach or transition to SPLE.

- *Experience curve and SPLE can be considered as alternative and complementary instruments to increase productivity*

Both the experience curve and SPLE are in fact different instruments to increase the productivity of software development. Experience curve utilizes productivity by learning from previous experience, while SPLE increases productivity by systematically reusing artifacts among similar products. Although these are different instruments to increase productivity, they can most certainly be used together to increase productivity. An SPLE approach in which the experience curve effect is high will indeed be the most beneficial. This information can be used by the management to invest in and plan activities for increasing the learning rate, for example, through pilot projects, courses, etc. This observation is also in alignment with the results of the systematic review in which the goals for adopting experience curve effect in software development was indeed to support the decision making and provide a more precise cost-benefit analysis.

- *The impact of experience curve effect gets lower in case the number of products increases*

The literature shows that the experience curve effect flattens in due time and its impact as such will reduce the more products are developed. This counts for both single system development and SPLE. The consequence of this is that for a sufficiently large number of similar products, even in case of an initial high experience curve effect, it will make sense in the end to choose an SPLE approach instead of single development.

- *The experience curve rate and the required upfront investment defines the strategy for product development*

It appears that both the experience curve effect and upfront investment define the result for ROI. The experience curve or upfront investment could be low or high. As such, we can identify different combinations of experience curve effect and upfront investment that might require a different strategy and decision. If the experience curve effect is high then single system development will usually perform quite well. In case of product line adoption the upfront investment is crucial. If the upfront investment is high together with a high experience curve effect in single development then a resistance to adopting SPLE can be expected. However, if the upfront investment is low and in case of a high experience curve effect then the adoption of SPLE becomes very attractive.

## 8. Related work

In our systematic review on experience curve effect in software engineering in particular we could not find any related secondary studies. Hence, the presented systematic literature review on experience curve effect in software engineering in this paper can be considered as a novel contribution. Our systematic review identified 15 papers that explicitly consider the experience curve effect and define various cost formulas that are used for different goals.

The second category of related work includes studies that discuss economic models to estimate the cost and ROI in software reuse. Several surveys have been published that identify, discuss and categorize these provided economic models for estimating the ROI of software reuse [32,33]. Within the context of SPLE several economic models have been provided that aim to estimate the ROI for transitioning to SPLE. In this context Ali et al. [7] provide an overview and comparative analysis of existing economic models to estimate the benefits of transitioning to an SPLE approach. For this, the authors use different perspectives including scope, types of analysis, economic function, underlying model, viewpoint, scenario, market attributes, cost factors, and risk adjustment. Based on the perspectives 12 selected approaches have been characterized.

In Table 9 we present the characterization of our approach with the characterizations of Ali et al. [7]. The scope of the economic models relate to either the adoption of SPLE or the entire life cycle. Our approach focuses on the analysis of both the existing single system development and the SPLE with experience curve. Hence we could state that the scope is primarily focused on analyzing the adoption of SPLE. Like most economic models the cost models that we have presented in this paper are used for cost and effort estimation. The basic economic function that we aim to calculate is the ROI, which seems also an important function in many economic models published in the literature. The existing economic models can be actually classified into basic models that are proposed from scratch and derived models that extend earlier defined models. Our approach is an extension of the earlier SIMPLE model [29] but in fact can also be applied to the other SPLE cost models with minor effort. The viewpoint of our approach concerns both the corporation and the system engineering. Since we adopt the SIMPLE cost model we also reuse the predefined scenarios. Like many economic models we do not concern market attributes and risks and uncertainties in the analysis process. We extend the cost factors of the SIMPLE model by integrating the experience curve factors.

The survey of Ali et al. includes the papers until 2009. We have elaborated on this survey and searched for other papers which were published after 2009. Based on this we could find three more publications [34–36] that discuss and present economic models in the context of SPLE. None of the papers in the survey explicitly address the notion of experience curve and do not integrate this in the cost models. However, the COPLIMO [37] model and its derivatives [38,35,36] implicitly assume the existence of the effect of learning and experience. The underlying models of COPLIMO and its derivatives is the COCOMO II model [39] that can be in principle used to model the experience curve effect. Several parameters of the COCOMO II model can be identified that relate to the experience curve. In this context, COCOMO II defines the so-called Personnel Experience (PREX) cost driver that combines the cost drivers application experience (AEXP), platform experience (PEXP), and language and tool experience (LTEX) which have range values

**Table 9**
Characterization of cost model.

| Perspective | Description |
|---|---|
| Scope | Adoption of SPLE |
| Type of analysis | Cost/effort estimation |
| Economic function | ROI |
| Underlying model | SIMPLE |
| Viewpoint | Corporate, systems |
| Scenario | Yes |
| Market attributes | No |
| Cost factors | $C_{Org}$, $C_{Cab}$, $C_{Unique}$, $C_{Reuse}$, experience curve coefficients |
| Risk adjustment | No |

from Very Low to Very High. Further, COCOMO II also includes the parameter UNFM in the adopted cost model, which defines the programmers' relative unfamiliarity with the software. The UNFM is measured using a simple Likert scale ranging from 0 to 1. These cost drivers could be in principle re-interpreted and calibrated specifically to model experience curve in the product line context. An interesting study is the integration of the experience curve formulas in the COCOMO II based models. In this context, Eickelmann et al. [26] have used the combined multiplier which is defined as the product of the COCOMO II cost drivers of APEX, LTEX, and PLEX in order to obtain a single learning curve model. Further, they have compared the results of this analysis with the results of the productivity learning curve that was defined in [27]. The authors seem to have considered single system development and did not discuss the impact of learning curve on SPLE adoption. This would be an interesting analysis which could be complementary to the approach that we have provided in this paper.

In fact, we could state that both the proposed explicit learning curve models and the parametric approaches have their own merits and could have been used to study the impact of the learning curve effect on SPLE adoption. The advantage of the explicit learning curve models is that these have been widely discussed in the management and education literature and very detailed studies have been published illustrating the usage of the learning curve models. Hence, these explicit learning models have a clear theoretical foundation from the perspective of the learning curve effect. In our case, extending the SIMPLE approach with the explicit learning curve models could also be integrated and adapted in a natural way. Because of the theoretical foundation behind both the SIMPLE approach and the learning curve models in the literature we could rely on the defined formulas for analyzing the learning curve effect on the SPLE adoption.

The advantage of the parametric approaches is that these are largely flexible and can be used to either reuse or re-interpret existing parameters or, if needed, new parameters can be defined. As stated before, these approaches could be used and enhanced to model the learning curve and analyze this for SPLE adoption. This may require, however, more effort than the straightforward learning curve effect models that already provide the required attributes and formula, which can be directly used for analysis.

Experience curve effect has also been discussed in software project dynamics [40] and software process dynamics [41] which deal with the dynamic nature of software process by using system dynamics simulation. According to Madachy [41], there are mainly three aspects: people applications, process and product applications, and project and organization applications. Experience curves are primarily discussed and related to people applications. There are also many studies related to learning curve and experience curve effect in other domains such as education, management science, and construction.

## 9. Conclusion

The impact of experience curve effect has been broadly discussed in management and education sciences. To the best of our knowledge a systematic literature on experience curve effects in software development has not been carried out before. The systematic literature review that we have carried out has shown that the experience curve has been addressed in several studies that focus on different aspects in software development. We have thoroughly studied all the identified 15 primary studies that resulted from the SLR and characterized and summarized these. Elaborating on these 15 identified primary studies and the literature on experience curve effect in management and education sciences we have presented refined cost models that integrate the experience curve effect in the cost models needed for estimating the transitioning to SPLE. These cost models have been provided for both single system development and software product line engineering to support a more precise cost estimation. In our analytical analysis process we have shown the impact of the various parameter values on the cost models with and without the experience curve effect. We could clearly observe the impact on the cost of development, time-to-market and the ROI. Our study has been further strengthened by applying the cost models within an industrial context. Perhaps the most important conclusion of our analysis is that the experience curve has a direct impact on both single system development and SPLE. This observation can support the decision making process in adopting and planning an SPLE approach. The experience curve can be also considered as a complementary instrument to SPLE to further increase the productivity. Our findings show that according to the adopted cost models and the corresponding analysis, adopting SPLE seems to pay off sooner or later. We have provided a useful and practical tool that can used and customized in software projects to support the decision making process. In our approach we have chosen to extend the SIMPLE model [29], because it is well-known and recognized in both the research and practice of software product line engineering. In this way we aimed to support the validity of our study. Complementary to our approach, modeling the experience curve using parametric approaches could be an interesting exercise to compare the results of both models. In our future work we aim to further experiment with the experience curve effect within an SPLE context and implement a decision support system that supports the presented refined cost models and the corresponding approach.

## Acknowledgments

This work has been carried out in Havelsan. We would like to thank Hakan Erdogmus for reviewing an earlier version of this paper and useful comments to improve the paper.

## References

[1] W.B. Frakes, Software reuse research: status and future, IEEE Trans. Software Eng. 31 (7) (2005) 529–536.
[2] P. Clements, L. Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley, Boston, MA, USA, 2001, p. 608.
[3] S. Buehne, G. Chastek, T. Kaköla, P. Knauber, L. Northrop, S. Thiel, Exploring the context of product line adoption, in: 5th International Workshop on Software Product-Family Engineering, 2004, pp. 19–31.
[4] K. Pohl, G. Böckle, F. Van Der Linden, Software Product Line Engineering: Foundations, Principles, and Techniques, vol. 49, Springer, Secaucus, NJ, USA, 2005, p. 467.
[5] K. Schmid, M. Verlage, The economic impact of product line adoption and evolution, IEEE Softw. 19 (4) (2002) 50–57.
[6] W.G. Sullivan, E.M. Wicks, C.P. Koelling, Engineering Economy, 15th ed., Prentice Hall, 2011.
[7] M. Ali, M. Babar, K. Schmid, A comparative survey of economic models for software product lines, in: 35th Euromicro Conference on Software Engineering and Advanced Applications, 2009, pp. 275–278.
[8] G. Bockle, P. Clements, J.D. McGregor, D. Muthig, K. Schmid, Calculating ROI for software product lines, IEEE Softw. 21 (3) (2004) 23–31.
[9] H. Ebbinghaus, Memory: A Contribution to Experimental Psychology, No, Teachers College, Columbia University, 1913.
[10] T.P. Wright, Factors affecting the cost of airplanes, J. Aeronaut. Sci. 3 (4) (1936) 122–128.
[11] J.D.J. Rodney, D. Stewart, M. Richard, Wyskida (Eds.), Cost Estimator's Reference Manual, second ed., John Wiley & Sons Ltd., New York, 1995, p. 744.
[12] B. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering, No. EBSE 2007-001, 2007.
[13] L.B. Raccoon, A learning curve primer for software engineers, Softw. Eng. Notes 2 (1) (1996) 77–86.
[14] K. Sherdil, N. Madhavji, Human-oriented improvement in the software process, in: Proceedings of the 5th European Workshop on Software Process Technology, 1996, pp. 145–166.
[15] K. Kang, J. Hahn, Learning and forgetting curves in software development: does type of knowledge matter?, in: ICIS 2009 Proceedings, 2009.
[16] S.A. Slaughter, J.A. Espinosa, Learning from experience in software development: a multilevel analysis, Manage. Sci. 53 (8) (2007) 1315–1331.

[17] S. Narayanan, S. Balasubramanian, J.M. Swaminathan, A matter of balance: specialization, task variety, and individual learning in a software maintenance environment, Manage. Sci. 55 (11) (2009) 1861–1876.
[18] N. Hanakawa, S. Morisaki, K. Matsumoto, A learning curve based simulation model for software development, Sci. Technol. 121 (19) (1998) 350–359.
[19] P.C. Pendharkar, G.H. Subramanian, An empirical study of ICASE learning curves and probability bounds for software development effort, Eur. J. Oper. Res. 183 (3) (2007) 1086–1096.
[20] P.K. Kapur, H. Pham, U. Chanda, V. Kumar, Optimal allocation of testing effort during testing and debugging phases: a control theoretic approach, Int. J. Syst. Sci. 44 (9) (Sep. 2013) 1639–1650.
[21] Y.a. Au, D. Carpenter, X. Chen, J.G. Clark, Virtual organizational learning in open source software development projects, Inf. Manage. 46 (1) (2009) 9–15.
[22] G. Abu, W. Cangussu, J. Turi, A quantitative learning model for software test process, in: 38th Hawaii International Conference on System Sciences (HICSS-38 2005), vol. 00, no. C, 2005, pp. 1–10.
[23] C.L. Huntley, Organizational learning in open-source software projects: an analysis of debugging data, IEEE Trans. Eng. Manage. 50 (4) (2003) 485–493.
[24] Y. Zorgios, O. Vlismas, A learning curve explanatory theory for team learning valuation, VINE (2009).
[25] M. Plaza, O.K. Ngwenyama, K. Rohlf, A comparative analysis of learning curves: implications for new technology implementation management, Eur. J. Oper. Res. 200 (2) (2010) 518–528.
[26] N. Eickelmann, A. Anant, S. Hyun, J. Baik, Quantitative control of process changes through modeling, simulation and benchmarking, in: Proceedings of the 17th International Forum on COCOMO and Software Cost Modelling, 2002.
[27] D. Dorenbos, A learning curve model for the quality and productivity of the software development process, in: Motorola Software Engineering Symposium, 1993.
[28] R. Madachy, People applications in software process modeling and simulation, in: Proceedings of the 6th International Workshop on Software Process Simulation and Modeling, 2005, pp. 160–163.
[29] P. Clements, J. McGregor, S. Cohen, The Structured Intuitive Model for Product Line Economics (SIMPLE), Technical Report CMU/SEI-2005-TR-003, No, February, 2005.
[30] F. Van Der Linden, K. Schmid, E. Rommes, Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering, Springer-Verlag New York Inc., Secaucus, NJ, USA, 2007, p. 334.
[31] M.A. Schilling, P. Vidal, R.E. Ployhart, A. Marangoni, Learning by doing something else: variation, relatedness, and the learning curve, Manage. Sci. 49 (1) (2003) 39–56.
[32] J.S. Poulin, The economics of product line development, Int. J. Appl. Softw. Technol. 3 (1) (1997) 20–34.
[33] W.C. Lim, Reuse economics: a comparison of seventeen models and directions for future research, in: Proceedings of the 4th International Conference on Software Reuse, 1996, pp. 41–50.
[34] A. Nolan, S. Abrahão, Dealing with cost estimation in software product lines: experiences and future directions, in: Proceedings of the 14th International Conference on Software Product Lines: Going Beyond (SPLC'10), 2010, pp. 121–135.
[35] A.J. Nolan, S. Abrahão, P. Clements, J. D. Mcgregor, S. Cohen, D. De, M. Hall, Towards the integration of quality attributes into a software product line cost model, in: Proceedings of the 2011 15th International Software Product Line Conference (SPLC '11), 2011, pp. 203–212.
[36] R. Heradio, D. Fernandez-amoros, L. Torre-cubillo, A.P. Garcia-plaza, Improving the accuracy of COPLIMO to estimate the payoff of a software product line, Expert Syst. Appl. 39 (9) (2012) 7919–7928.
[37] B. Boehm, A. W. Brown, R. Madachy, Y. Yang, A software product line life cycle cost estimation model, in: Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE '04), 2004, pp. 156–164.
[38] H.P. In, J. Baik, S. Kim, Y. Yang, B. Boehm, A quality-based cost estimation model for the product line life cycle, Commun. ACM 49 (12) (2006) 85–88.
[39] B.W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, R. Madachy, B. Steece, Software Cost Estimation with Cocomo II, first ed., Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.
[40] T. Abdel-Hamid, S.E. Madnick, Software Project Dynamics: An Integrated Approach, Prentice-Hall Inc., Upper Saddle River, NJ, USA, 1991.
[41] R.J. Madachy, Software Process Dynamics, Wiley, 2008, p. 601.