6

Cost Estimation for Independent Systems Verification and Validation

András Pataricza[1], László Gönczy[1], Francesco Brancati[2], Francisco Moreira[3], Nuno Silva[3], Rosaria Esposito[2], Andrea Bondavalli[4,5] and Alexandre Esper[3]

[1]Dept. of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary
[2]Resiltech s.r.l., Pontedera (PI), Italy
[3]CRITICAL Software S.A., Coimbra, Portugal
[4]Department of Mathematics and Informatics, University of Florence, Florence, Italy
[5]CINI-Consorzio Interuniversitario Nazionale per l'Informatica-University of Florence, Florence, Italy

Validation, verification, and especially certification are skill and effort demanding activities. Typically, specialized small and medium enterprises perform the independent assessment of safety critical applications. Prediction of the work needed to accomplish them is crucial for the management of such projects, which is by nature heavily dependent on the implementation of the V&V process and its support. Process management widely uses cost estimators in planning of software development projects for resource allocation. Cost estimators use the scoring of a set of cost influencing factors, as input. They use extrapolation functions calibrated previously on measures extracted from a set of representative historical project records. These predictors do not provide reliable measures for the separate phases of verification, validation and certification in safety critical projects. The current chapter

summarizes the main use cases and results of a project focusing on these particular phases.

6.1 Introduction

Verification and validation (V&V), and Independent Software and System Verification and Validation (ISVV) are crucial in critical system development. However, the execution of such activities underlies strict time and budget limits and depends on input elements delivered by the entity performing the design and implementation of surprisingly changing quality.

Our research (described in details in CECRIS [1, 2]) focused on synthetizing a method for estimating the cost and focusing the effort of V&V activities in order to make these critical steps more managed and foreseeable in terms of time and cost aspects.

This necessitated an appropriate, organized, and scientifically well-founded working methodology.

The traditional software and system design industry has well-elaborated methods for the assessment of the aspects of efficiency and quality w.r.t. the human actors, technology used and project management background of the companies. Cost estimators (CEs) use the scoring of a set of cost influencing factors, as input. Their respective predictor uses extrapolation functions calibrated previously on measures extracted from a set of representative historical project records.

6.1.1 ISVV Workflow

The domain standards [3–8] define the project lifecycle and forms part of the design workflow and its phases. They envisage a purely top-down process starting with the requirements for implementation and checks. The individual phases are self-contained in the terms of design and V&V. Figure 6.1 presents a schematic view on the interoperation of the development and V&V phases.

However, the practice indicates, that in many cases there are deviances of this idealistic model. For instance, changes in the specification triggered by the end user during the design may result in iterations.

Constraints related to delivery time force occasionally corrections executed in parallel in phases intended to be sequential by the idealistic process model. We suppose a typical "waterfall" development, which, however, has its disadvantage that decisions taken in the project are often not

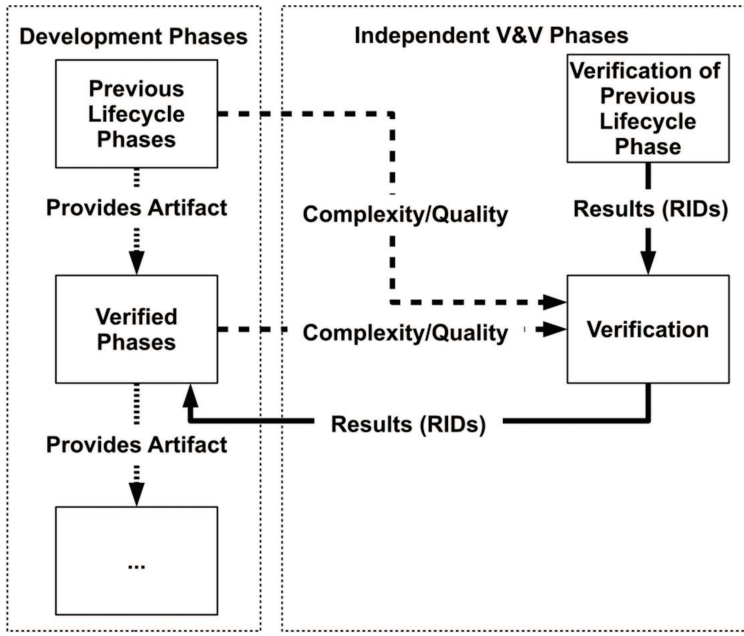

Figure 6.1 Schematic view on V&V activities.

reflected by artifacts (e.g., design documentation) created in an earlier project phase.

Even in critical projects, due to resource constraints, early deliverables are often not updated which may result in inconsistent documentation.

Note that the Figure 6.1 shows a simplified partial view of the process presented in Pataricza et al. [9].

Verification and validation activities of a particular phase and design activities in the successor phase frequently show a similar temporal overlap, for instance, coding starts before the completion of design verification.

If design verification detects faults later, feedforward change management has to take care to correct the design and update the specification of the already ongoing coding activity.

Furthermore, latent faults escaping the V&V of their respective earlier phase trigger feedback loops and multi-phase correction actions affecting preceding project deliverables as well.

Review Item Discrepancy (RID) is the output measure of a V&V activity. A RID is an issue, identified by a reviewer who is uncompliant with a requirement, a review objective or a design goal.

Our suggestion is to facilitate effort estimation and "spot out" problematic parts of the target artifact of verification by applying complexity/quality metrics. These metrics can be retrieved right at the beginning of verification; also, such metrics for previous project deliverables (e.g., code metrics in the case of test verification) are reusable.

6.1.2 Objectives

Cost estimators are fundamental elements all along the lifecycle of a project from the proposal definition through the project execution and *a posteriori* evaluation. Their main purpose is assuring a timely execution of the target project without overspending. A variety of general-purpose CE methodologies exists supporting the project management activities by predicting Key Performance Indicators (KPI), such as time/effort/cost/quality/risk.

Similarly, different measures assure the compliance of an ISVV or certification process with the standards. However, the objective of an assessor is an effective V&V and certification process among the standard-compliant ones. Effectivity means here both productivity and quality.

Rough estimates, and rules of thumbs like approximating testing related costs as by around 40% of the overall development project costs, as typical for non-critical applications do not deliver reliable results for safety-critical ones.

Academic experts performed a thorough going evaluation of the current practice of the ISVV companies performed for third parties. They analyzed numerous historical project logs with a particular focus on the cost aspects to identify the main factors influencing the efficiency of the projects and the dominant quality and productivity bottlenecks. One main observation was that automatic quality checking of incoming project artifacts has a high priority, as this is the core factor determining the efficiency of the entire assessment projects.

The primary objective of the research based on the results of the analysis was checking the validity of the original approach of creating general-purpose CEs for the purpose of ISVV-related effort prediction.

The most important use case is the improvement of the ISVV process. Accordingly, the focus of the evaluation was on the "what-if" – like analysis of the impacts of factor-by-factor changes in the process factors. This way, sensitivity analysis can predict the relative improvement expected, for instance, by the introduction of a new technology.

6.1.3 Approach

The enrichment of the ISVV workflow design by effort metrics necessitates a proper cost predictor. The literature refers to a large number of general-purpose and dedicated approaches.

Flexibility, understandability and possible accuracy were the main selection criteria in designing a CE specific for V&V.

The COCOMO family of CEs served as the starting point, as this is popular and relies on an open model. Note that the different versions of the CEs covering different use cases take nearly twenty factors as input to generate a cost estimate (as detailed in the next section).

The moderate number of historical project log does not even adequately sample this vast parameter space allowing the generation of an entirely new CE dedicated to ISVV.

Accordingly, our approach consisted of the following main steps:

- estimation of those general factors related to software development processes which are independent of the peculiarities of ISVV;
- cross-validation of the measured predicted by the general-purpose CE and the logged efforts in the ISVV sample set;
- checking the set of input factors for completeness and potential revision of the definition according to ISVV.

6.2 Construction of the ISVV Specific Cost Estimator

Software project management offers a huge variety of approaches and tools for cost estimation. As no single one is detailed enough to describe the specific V&V processes related to critical systems, we started the elaboration of an ISVV-specific CE model referred further as CECRISMO [10].

The COCOMO family of CEs served [11–13] as the starting point of the designated CECRISMO. The COCOMO family of CE has a widespread industrial acceptance and use including (embedded) system design. Members of it support different project types and process models including mixed hardware–software development, component reuse, and integration-based system design, etc. All COCOMO styled predictors rely on an open model. Thus, the adaptation can follow the usual process of the creation of a new member of this family. Most members of the family have an open source tool support.

6.2.1 Structure of the Cost Predictor

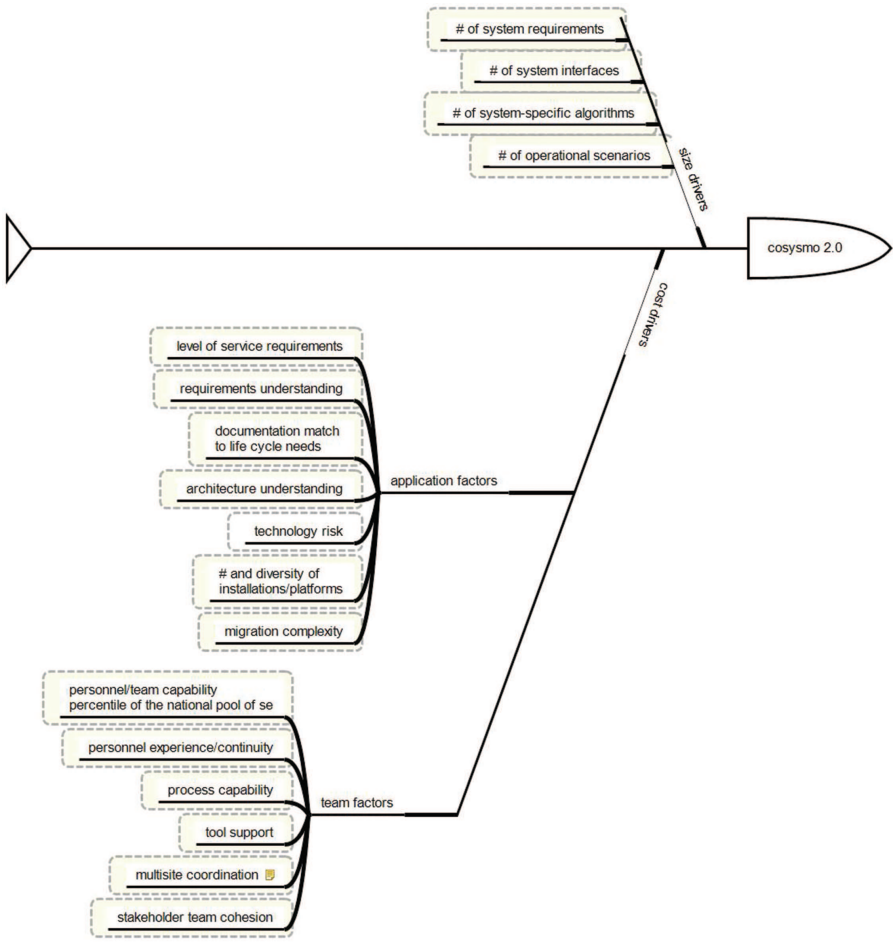

Figure 6.2 COSYSMO 2.0: Size Drivers/Effort Multipliers.

All the members of the COCOMO family (incl. COSYSMO) share a similar formula for cost prediction (see Figure 6.2).

$$\text{Person Months}_{\text{Nominal Schedule}} = A(\text{cost driver})^E \prod \text{effort multiplier},$$

where

- $Person\ Months_{Nominal\ Schedule}$ is the estimated effort in man-months needed for the end-to-end execution of the project (nominal schedule),

- "$size\ driver$" is an estimated metrics (a single real number) of the size and complexity of the target of the project ("*What is the output of the development project?*").
- For instance, early COCOMO cost predictors used the estimated number of source line of codes or function points as size estimators in pure coding projects. As not all elements within a particular category result in the same amount of efforts, difficulty categories express the differences in efforts and the total numbers are calculated as weighted sums. These weights are in the range of one order of magnitude for each step of a difficulty category. Similarly, the decreased efforts due to reuse of already existing artifacts is weighted by a reduction factor.
- "*effort multiplier*" expresses the efficiency of the development process from technology, skills, development infrastructure and organizational aspects ("*How effective is the output of the development project elaborated?*");
- The evaluator expert assigns to each individual effort influencing factor a grade (an ordinal measure) **Very low/Low/Nominal/High/Very high,** which in turn is converted with an aspect dependent empirical constant vector into a relative effort multiplier (for instance an effort multiplier metric less than 1 corresponds to a speedup w.r.t. to the nominal case).
- A and E are calibration constants estimated during curve fitting to the data in the calibration set.

6.2.2 Cost Drivers

COSYSMO targeted the entire system development process in the large. When designing a new member of the COCOMO family, the standard procedure is an analysis and if necessary of the original model simultaneously keeping the core of its original factors and prediction methodology.

The following evaluation targets the creation of a CE corresponding to ISVV in focus. This analysis addresses the following central questions:

- How appropriate are the input factors defined originally for the end-to-end process for ISVV (completeness and scoring methodology)?
- Is there a necessity for the re-interpretation of the factors originally defined for end-to-end development to the peculiarities of ISVV?

6.2.3 Focal Problems in Predicting Costs for ISVV

The primary reason of the non-existence of ISVV specific CEs is the lack of a sufficient number of available project logs and other processable artifacts supporting statistically justified generalized statements.

In addition, the efforts needed for ISVV heavily depend on the quality of the input artefacts supplied by the separate design entity. This way, the input quality has an at least equally important influence, like the size and complexity of the system under assessment.

These important interdependencies necessitate a re-interpretation of the individual factors.

General-purpose CEs have to cover a broad spectrum of human skill levels ranging from a post-secondary level programmer to domain experts. Accordingly, they apply a rough-granular scoring range covering this wide set. ISVV, a highly demanding task is typically carried out by SMEs having a well-educated staff frequently having a higher academy degree and a long industrial expertise. This way, the staff in ISVV belongs to the top few scores related to human factors in general-purpose CEs, and the expressive power of these factors becomes insufficient due to the low resolution.

6.2.4 Factor Reusability for ISVV-Related CE

The **size drivers** express the scale of the system under development adequately, but the size alone is insufficient to determine or approximate the efforts necessary for ISVV.

In the case of system requirements and artifacts corresponding to them, the impact of safety criticality on ISVV-related efforts needs refined weighting methods and scores. Critical requirements, components, etc., necessitate a thoroughgoing analysis as defined in the standards.

Moreover, standards define variation points in the checking process. Accordingly, here the weight of the factor needs a fitting to the actual process instance chosen for the actual object of ISVV instead of a global weight used in the original CE.

Similarly, as ISVV is a follow-up activity of a design phase the design quality heavily influences the amount of work to its execution.

The quality of the outputs of some V&V preliminary activities (e.g., design the V&V Plan, Requirement Verification, and Hazard Analysis) are major effort drivers for rest of the V&V cycle. Modeling this behavior will also allow to perform ROI analyses at early stage of the V&V process and/or continuously monitoring the cost-quality tradeoff of the overall V&V cycle.

The notion of quality has a double meaning here. On the one hand, it refers to the care of the preparation of the ISVV input artifacts (how well is a design document structured and formulated). The quality of the input artifacts

has a direct impact on the problem size (how many items needs the review report to detail).

On the other hand, it covers technical aspects (like the testability of code). The expressive power of the grading system is insufficient, and this issue is subject to a detailed analysis in a subsequent section.

6.2.5 Human and Organizational Factors

An important obstacle originates in the difference regarding the organizational background.

Companies developing non-critical applications typically follow an end-to-end in-house approach. Mono-company development assures independence at the team-level within the company boundaries, if required by the standards, at all. Both the development and V&V teams share the same enterprise culture regarding skills, organizational and technology aspects in case an entirely in-house process. This way the assumption of having a nearly homogenous culture and quality along the entire process is highly probable. Experts performing the scoring of the individual organizational and technology-related cost factors can typically do it at the level of the company.

On the contrary, ISVV relies on two organizationally independent, but to a given extent collaborative partners each having his separate culture. Moreover, ISVV performed for different clients faces different cultures at the side of the designer companies.

The group of team factors has nearly the same semantics as in the original model.

However, ISVV is typically a much more complex activity using different methods and tools as a traditional development process. This way, for instance, human and technical factors are related to the individual activities instead of using flat, global grading. If an ISVV process consists of a mix of peer review and formal analysis, tool support, personal experience, etc. all have to be evaluated at the resolution of these individual activities instead of a single approach based scoring. Mostly SMEs having a limited number of experts perform ISVV. Occasionally, the proper level of assessment is that of individuals instead of "average programmer at the company."

The evaluation of cooperation related factors needs adaptation, as well (multisite coordination, stakeholder team cohesion). These aspects cover namely two kinds of teams: Intra-team communication at the unit performing ISVV where the interpretation is identical with that in the design organization; inter-team communication between the design entity and the ISVV site. A good approximate solution is a simple logic taking a pessimistic approach

by assigning the minimum score of the two particular scores to these two communication-related aspects.

6.2.6 Motivating Example: Testing

One of the core issues is the dependency of ISVV related costs on the quality of the input artifacts (like software source code), while the output of the process has to comply with the strict quality requirements formulated in the standards.

This dichotomy in the requirements has non-trivial consequences on cost estimation.

- Testing of non-critical applications has as input typically a moderate quality code. It has to reach an acceptable, but not an extremely high fault coverage.
- Testing of safety-critical applications however, always targets high-fault coverage.

According to numerous observations, the detection rate drastically drops along the progress of the testing process after an initial peak (the testing time to detection rate distribution usually corresponds to a long-tailed Rayleigh distribution as shown in Figure 6.3, see [14]).

- This way effort estimation (the time to finish testing by reaching the required coverage) has to cope in case non-critical applications dominantly with the initial bulk of faults (which is relatively easy to estimate).
- Regarding safety-critical objects under test this termination time instance lies on the long flat tail resulting in a potentially large estimation error.

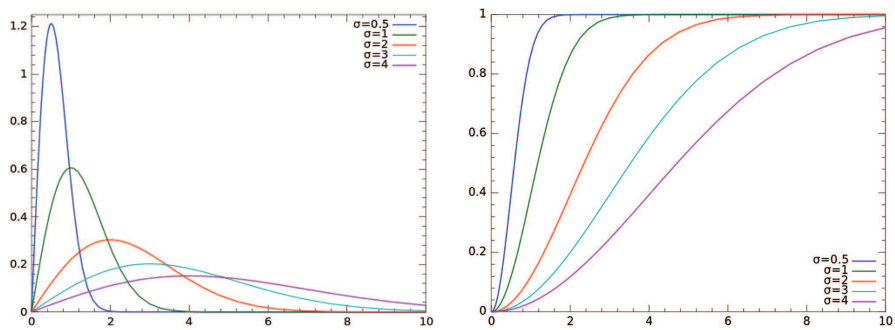

Figure 6.3 Rayleigh distribution by different parameters (a) fault detection rate (b) fault coverage (source: wikipedia.org).

This way the scoring scheme of traditional CEs is inappropriate, as a score corresponding to a "High input quality" would mean, that fault coverage is above a threshold. However, this corresponds in testing time to an interval between the threshold and infinity.

However, input characterization may improve estimation accuracy by substituting the original grade-like scores with continuous metrics. We used mainly visual Exploratory Data Analysis methods in order to find and validate main correlations and interdependencies on project data. During CECRIS, some typical and "atypical' projects were examined in order to see what connections and measureable characteristics can be set up between outputs and inputs of ISVV activities.

6.3 Experimental Results

Even the initial experiments delivered some highly valuable conclusions for prioritizing the individual goals in CECRIS.

Note that the low number of cases presented here is apparently insufficient to draw any conclusions with a sound mathematical foundation but they serve as an orientation for the further experiments.

6.3.1 Faithfulness of the Results

Figure 6.4 shows the effort estimated by COSYSMO compared to the real effort spent by the V&V activities for each pilot project.

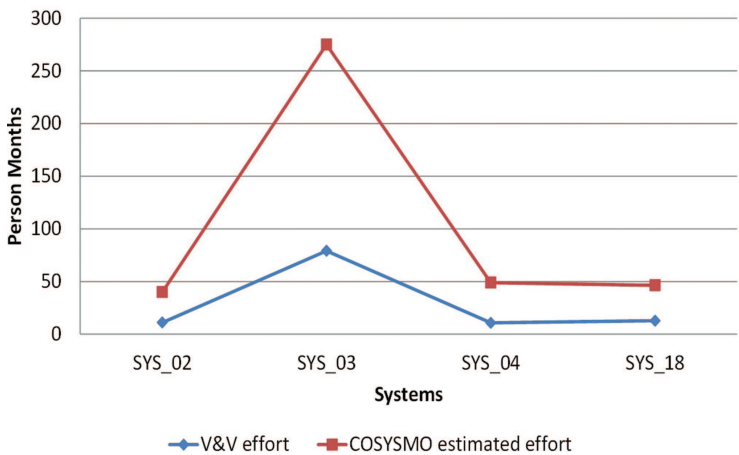

Figure 6.4 COSYSMO estimation compared to real V&V effort.

The ratio of the logged and predicted efforts is for the entire four projects $4 \pm 10\%$. The correlation coefficient is 0.99, which means a strict proportionality. (It is well visible in the figure that the two curves have a similar shape.)

This way, the first impression is that the direct application of COSYSMO to pure V&V phases delivers a qualitatively correct, but in absolute numbers wrong estimation. The main reason is that COSYSMO accounts for the efforts of the entire development process (starting with requirement analysis and including the entire code development and in-production testing phases, as well). The reference values from project logs cover only a smaller set of sub-activities of this process, V&V, accounting for 22–28% of the global project efforts.

The ratio of V&V efforts w.r.t. to global efforts is in a narrow interval over all the pilot projects.

Expert's estimations predict in general 30–35% share for V&V in the industrial practice depending on the project and product in contrary to the observed $\sim$25% in the pilot calibration set. This better than average productivity originates in the very highly skilled and experienced personal typical in university close spin-offs.

Note that while the observed $\sim$25% holds in the pilot calibration set, but this value should be further validated on a larger data set. It presents a further problem that the independent V&V and certification organizations have no exact data on the ratio to the total effort, while CECRISMO explicitly estimates effort for these phases.

COSYSMO can serve this way as a comparison base if the ratio of a particular V&V and certification activity can be properly approximated.

V&V efforts can be taken as a constant fraction of the predicted total development costs if only a rough estimate is targeted. Some COSYSMO implementations rely on the assumption on a constant ratio, which is valid if the entire project is carried out completely in-house.

However; the assumption has globally no proper justification in the case of an external accessor. The total effort estimation mixes activities carried out at the product manufacturer (3/4 share) and V&V carried out at the independent accessor. A rigid subdivision cannot properly predict V&V related efforts in general, as all factors may differ in the two actor enterprises.

However; such a first estimate is a candidate for a rough relative comparison of two solutions of V&V tasks carried out by the same company.

6.3.2 Sensitivity Analysis

Sensitivity analysis is widely used in order to estimate the impact of changing some input value or values onto the output of a function or a system.

The corresponding inputs are the size drivers and the effort multipliers in the case of cost estimation.

Sensitivity analysis forms the basis of answering numerous critical questions:

- **Impact prediction for process changes**: Process improvement necessitates always investments like the introduction of new tools into **technology**, improvement of team cohesion by teamwork support or investment into the human capital by training. While it is generally true that such investments has a beneficial impact onto the productivity and cost, *sensitivity calculation is able to predict their impact in quantitative terms*, as well.
- **Adaptive project management**: Sensitivity to the size drivers is a main characteristic to describe scalability of a project. Here the breakdown of the different size drivers into qualitative categories and taking *the individual sensitivity values for instance delivers a good indicator of system specification modifications during the process*.
- **Estimation of the impacts of misscoring**: The different effort multipliers need a special care from the point of view of sensitivity analysis. These are all categorical variables with assigned ordinals as domain (the effort multipliers take their score values from an ordered enumerated list).

By the very nature of the scoring by an expert judgment guided by an informal description is somewhat subjective.

The evaluator can score a particular factor by one score up or down. *Sensitivity analysis here answers the question what are the impacts if the evaluation expert puts a score to a wrong value* (typically some of the neighboring values of the proper one).

The estimation of the impacts of such scoring errors assures that a range of uncertainty can be provided in addition to the single effort estimate as the final result. (Some cost estimation methods apply a non-deterministic simulation around the expected scoring to deliver such an uncertainty estimate.)

The V&V focus of CECRIS necessitated this analysis, as the focus of the original COSYSMO description was not completely identical with

the CECRIS objectives; this way, miscategorizations may occur. Another problem was that the scope of COSYSMO is essentially wider than that of CECRIS. This way, a significant part of the large domain of effort multipliers is irrelevant in the cases of CECRIS (for instance, no staff of very low skills and novice to certification will be involved into V&V activities despite the fact that a this is a valid score allocation in COSYSMO).

Accordingly; only a subdomain of the COSYSMO's score space is relevant for CECRIS and this raises the question of applying a narrower but finer granular set of candidate scores.

A sensitivity analysis of all the individual cost drivers was performed in order to estimate the potential impact of miscategorization and a refined scoring system.

This is basically performed by taking the nearest lower and higher score of an input factor. If the mismatch between the observed value and the predicted one drastically changes, thus the cost is highly sensitive to the particular factor than the scoring rules have to be revised.

If the observed values typically lie between the two predictor values corresponding to neighboring score settings, than a refinement of the input scoring (introduction of an intermediate value) may increase the resolution and accuracy.

The Figure 6.5 shows the relative impact of three cost drivers, **Requirement Understanding**, **Architecture Understanding,** and **Personnel Experience/Continuity**, on the cost estimator. Only these three cost drivers are analyzed for the sake of simplicity and without lose generality. As shown in Figure 6.5, increasing cost driver scores (x-axis), decreases the cost estimate (y-axis) nearly in a linear way. **Requirements Understanding** has a higher impact on the cost like **Architecture Understanding** and **Personnel Experience/Continuity**.

Even this small example indicates the importance of sensitivity analysis from the point of view of project- and process management.

If the quality of the requirement set specification is improved by a score than $\sim$20% can be reached in the terms of cost saving. If for instance, the traditional textual specification is substituted with an unambiguous formal model in an easy to understand form like the mathematically precise but well readable controlled natural language, then a significant cost saving can be reached with relatively low effort in training.

Naturally, multiple cost factors change in the case of the introduction of a new technology.

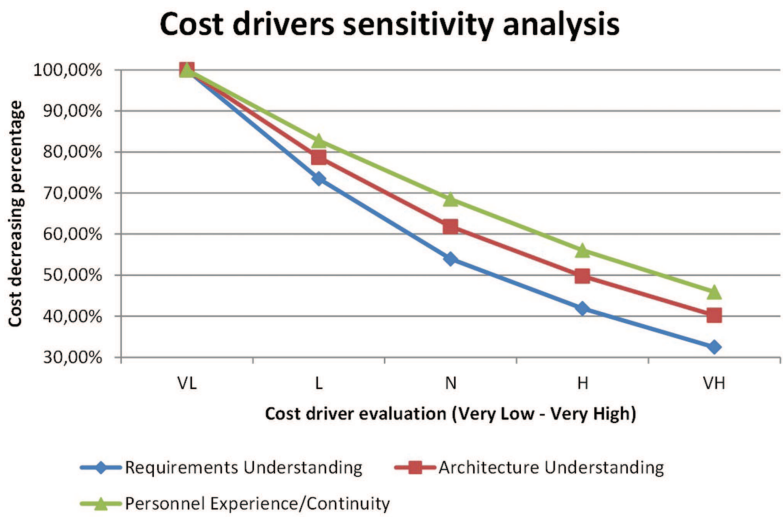

Figure 6.5 Cost drivers sensitivity analysis.

6.3.3 Pilot Use Case for Project Management

As pilot case for what-if analysis the core CECRIS action line was selected: In a company dealing with V&V/certification of critical ES advanced academic approaches are introduced. The skills of the local personnel would be at the beginning moderate and may only gradually reach the level of professional expertise. As a special case we investigated what happens if intensive coaching was provided by senior experts to help the transition. One of the previously analyzed projects was selected to carry out the what-if analysis.

COSYSMO calculator was taken and factors were determined as summarized by Table 6.1. Concentrating only on factors (taking values from Very Low through Nominal to Very High) where significant impact is expected, leaving other multipliers unchanged.

Our pilot calculation showed that a project introducing formal methods without experience ("Introductory") has approximately the same cost as a

Table 6.1 Pilot use case for introducing formal methods in verification

Factors	Orig.	Introductory	Final	Intermediate
Requirements understanding	H	N	VH	VH
Personal experience	N	L	N	N
Tool support	VL	H	VH	H

H, high; L, low; N, nominal; VH, very high.

project working with traditional, mostly human checks. Due to increased uncertainty, real costs (and technology related risk) may even be higher in this case. On the other hand, a guided "coaching" may result almost the same cost/effort saving as the ideal "final" stage, due to the reduced risk of novel technology and lack of experience (50 vs. 60%). Although these factors were not primarily calibrated for V&V (nor for embedded systems), such overall project management considerations also hold in this area.

6.4 Case Studies

In this chapter we present case studies to illustrate how the estimation method can be used on real-life data. During this analysis, we were also validating our assumptions about the process model.

6.4.1 Complexity Factors

Complexity is an essential factor, used both in COSYSMO and in *ad-hoc* effort estimations for ISVV. As we described in Section 6.1, one overall scoring for projects cannot describe the overall difficulty of the problem, therefore a more fine-grained estimation is needed. Scaling/rating of the input may also depend on the problem or domain, so this estimation should be calibrated to the domain and the nature of the problem as well (e.g., boot software, a communication submodule or a computationally intensive software component may be different). Note that this complexity is expressed also in the COCOMO family as "algorithmic complexity". In case of ISVV, we can rely on the advantage that the software to be checked is already available, so metrics which are typically measurable on the outcome of a traditional development project can be used for input characterization.

Although the interpretation and calculation of complexity differs across domains and development steps, we took some examples which are easy to calculate. Here we introduce complexity in the sense of code complexity.

Code complexity is measured usually on the source code of components. It concentrates on measuring the structure of the code (which do not necessarily correlates with the actual complexity of the problem solved by the particular software component, which is the main criticism wrt. complexity measurements).

One of the most widely used metrics is McCabe complexity:

$$\#\boldsymbol{of\ linearly\ independent\ paths\ in\ CFG} = \\ \#\boldsymbol{Edges} - \#\boldsymbol{Nodes} + \mathbf{2} \times *\#\boldsymbol{ConnectedComponents}$$

This metric fits well with problems which are control dominated, captured by the structure of their Control Flow Graph (CFG).

The above definition is independent from the programming language but is influenced by the coding convention, therefore its application may be limited for domains/teams. It should be quantized (e.g. low-medium-big). A typical upper threshold for "low" complexity may be five for instance in embedded software.

Another important metric is the *Halstead* metric concentrating on the "language" of the software, measured in the number of terms used.

Halstead metrics are concentrating on the "size" of the software measured in the number of "operators" and "operands" in software.

Let n1 be the number of *unique operators* and n2 the number of *unique operands*, while N1 denotes the *total* number of *operators* and N2 stands for *total* number of *operands*.

$$\textbf{\textit{Vocabulary(n) = n1 + n2}}$$
$$\textbf{\textit{Length(N) = N1 + N2}}$$
$$\textbf{\textit{Volume(V) = N *log}}_2\textbf{\textit{n}}$$
$$\textbf{\textit{Difficulty(D) = (n1/2)*(N2/n2)}}$$
$$\textbf{\textit{Effort(E) = D*V}}$$
$$\textbf{\textit{Delivered Bugs(B) = V/3000}}$$

It is important to note that the latter two derived metrics are highly domain-dependent. Effort refers to actual effort creating the software while "Delivered Bugs" is a rough estimation of "faults inserted". E.g., the number of Delivered bugs is known to be an underestimate in general-purpose C/C++ programs but is in the range of ISVV found bugs, since ISVV deals with "residual" bugs not detected by in-house testing and other means of fault detection.

The programming language itself has a trivial impact on these metrics, especially the Halstead length of a program.

These metrics can also be applied at the model level since most of their concepts are already captured by (detailed) software models. Similarly to McCabe metrics, Halstead (derived) metrics can be used to calculate size drivers for COCOMO-like estimations.

We have recently experimented (among others) with C, Java, Python, JavaScript and found that these metrics may be used on only for input characterization, but also for finding outliers. For instance, in the examined subset, there were different solutions available for a computationally intensive

problem written in the same programming language with drastically different complexity metrics. It turned out that one naïve solution is inefficient, but simple iteration logic while the other solution relied on specialties of the problem. This second component would be trivially harder to maintain but runs with less execution time for the majority of the inputs. However, this is such a difference which could not have been found only by looking at the requirements or test cases of the given software component.

These examples also illustrate that the COCOMO family is too high granular when talking about factors, basic estimations should be supported by more detailed metrics. Most of the effort multipliers, however, will remain similar. These methods can also be used to select the appropriate level and target of the application of (semi) automated formal methods.

6.4.2 Cost Impact of Requirement Management

Taking the example of a breakdown of a difficult requirement to five simple ones, the difference is 50% (2.5 vs. 5) in nominal difficulty. Of course, these numbers represent more rules of thumb than exact calculation, but still they express industrial best practice in project planning. Structuring of requirements is therefore a crucial part of project design. Although this can be done in any textual tool or in Excel (the most widespread requirement definition tool), a structured, object-oriented approach can help requirement refactoring and reuse by introducing typed interdependencies and domain specific notation in requirement specification.

As requirements have a specific "lifecycle", changes and modifications have significant impact on overall project difficulty, and therefore affect related ISVV activities.

Table 6.2 shows, for instance, that a requirement of "Nominal" difficulty has a 30% less effort implications if it is reused from an existing requirement set.

Table 6.2 Effect of requirement lifecycle

No. of System Requirements	Easy	Nominal	Diff.
New	0.5	1.0	5.0
Design for Reuse	0.7	1.4	6.9
Modified	0.3	0.7	3.3
Deleted	0.3	0.5	2.6
Adopted	0.2	0.4	2.2
Managed	0.1	0.2	0.8

Requirements here are measured in a uniform way, however, from safety critical point of view, there can be huge differences even between requirements of same category (e.g., change in a "nominal" requirement related to emergency shutdown may have larger impact on software testing).

Also there are interdependencies among the requirement set, which may override the above numbers.

6.4.3 Automated Analysis for Factor Selection

Besides exploratory analysis methods, a wide selection of automated analysis technique is available in "algorithm as a service tools" like IBM Watson Analytics, Microsoft Cortana or other evolving services. These tools typically support data cleansing and evaluation by combining a selection of well-known statistical algorithms with heuristics and other (e.g., text mining based) methods in order to find correspondences between input variables of datasets.

These tools return with a number of suggestions which in turn can be justified, refined (or even invalidated) by a profound analysis and check of human experts. These automated methods can not only speed up the initial steps of data analysis, but also systematically reduce the chance of overlooking factors or biasing analysis by false assumptions.

We analyzed 7 different ISVV projects with source code files in the range of 500 from the aerospace domain and submitted input information (complexity measurements, number of requirements/files to check, etc.) and output evaluation (number of RIDs found, information about these RIDs w.r.t. input artifacts, overall effort estimation) and performed automated analysis in order to see what correspondences and implications are derived. We were using IBM Watson Analytics. Our findings were the following:

Input quality. As expected, data quality was very heterogeneous across projects. The tool pointed out factors where the number of missing entries (NAs) or constant data may distort the analysis. Input quality information is important as filtering out irrelevant or partially missing factors can prevent later analysis methods from generating false/not interpretable results.

Categorization. Without any previous domain knowledge, some interesting categorization suggestions (expressed in the form of automatically derived decision tree) were found. The categorization returned by the tool was approximately the same as returned by experts. Such information might be used in qualitative "labelling" of projects.

Heat map on frequent combination of factors. When selecting input factors, it is important to know which combination of values (intervals) appear in data to see how realistic/relevant the scaling of factors is. Analysis methods returned some important suggestions, for instance on how to combine readability metrics and file size to find a rough estimator for input quality (fault density measured in the "frequency" of RIDs).

Derived factors. The analysis tool was also able to derive estimators (based on partial linear regression) which can take the project characteristics into account and return function-like closed formula predictors where parametrization may depend on project nature (which may be derived from the same metrics as used for project categorization).

Of course, the above estimations/predictions do not hold for all V&V projects, but such automated analysis results may speed up the domain specific estimation and quality improvement process by inserting systematic analysis into today's mostly ad-hoc method.

6.4.4 Quality Maintenance Across Project Phases

In order to see the effects of multiple-phase development (and, therefore, multiple-phase ISVV activities), we took two examples from the aerospace domain [9].

Our research questions were the following:

- How does the "coding style" (i.e., structuring the code in smaller/bigger files with corresponding "header/specification" information) affect the quality of the code?
- How do the number if iteration and the timespan of the project (and, therefore, the ISVV activities) influence the quality?

As Figure 6.1 shows, quality of artefacts may affect subsequent project phases, and thus, the input of the corresponding V&V activity. In critical projects, traces among phases are typically available and support the identification of engineering decisions (e.g., how many classes will implement a certain requirement).

We also tried to introduce metrics to support the evaluation of dependency between phases in a quantitative way; the number of requirements/KLOC is such a candidate.

Main findings of this experiment were the following:

- Although traceability is assured by the development process and tools, the effect of a fault inserted in an early phase (e.g., in a requirement) is not always corrected, especially during re-iteration.

- Mainly due to the mostly human work during verification phases, RIDs are often recorded in a way which does not help later analysis: typical faults (e.g., in coding style, comments, or even implementation not fully consistent with design documents) may be reported once but may refer to multiple artefacts. Moreover, this also underlines that the number of RIDs may not be a precise characterization of ISVV project output, since the granularity of RIDs might move on a wide scale.
- The timespan of the original project heavily influences the "lifecycle" of artifacts and also the quality assurance. We concentrated mostly on completeness and correctness faults, as during the re-iteration phase of ISVV, these get a special emphasis. We measured the fault ratio per KLOC in the source code. In the case of an "incremental" project with shorter iteration times (see the left part of Figure 6.6), the project converges and the ratio of faulty input items drastically decreases. In the case of long iterations and multiple changes in requirements, both the completeness and correctness of the software code got significantly worse. Note that these relatively small fault rates (when compared to a rule of thumb of 5 faults/KLOC) are detected after the code has been approved by in-house checks.

This experiment also underlines that measuring only the characteristics of input artefacts is not enough without the organizational and human factors,

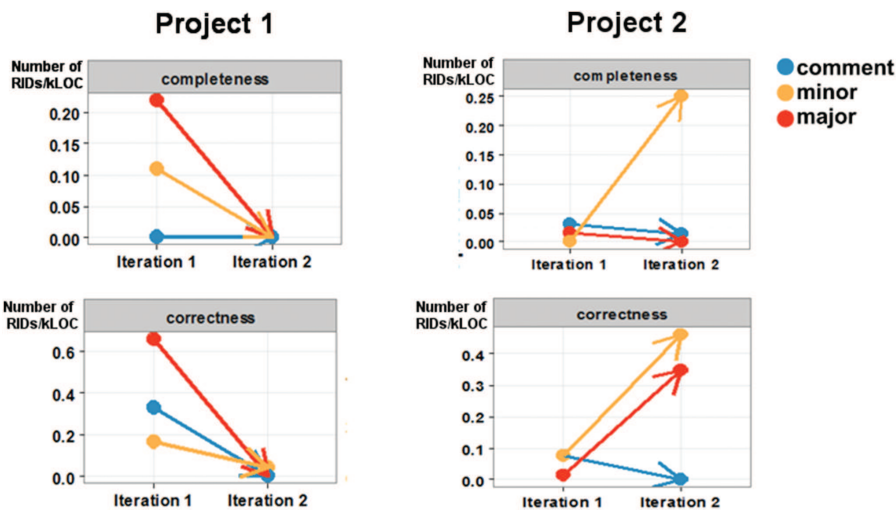

Figure 6.6 Trends of fault in multi-phased ISVV projects.

which obviously have a huge impact on project effort both at the customer side and at the ISVV company.

6.4.5 Fault Density and Input Complexity

In the following section, we present an analysis example which tries to set up high level correspondence between fault density (measured *post-mortem*, at the end of closed ISVV project phases) and complexity metrics (measurable at project planning stage).

A simplified and cleaned set of data has been loaded to Microsoft Power BI tool used for data analysis and visualization. Results presented here do not depend on the particular tool of choice. Besides basic "BI" visualizations, some R code was used to generate the plots. A part of the dashboard is shown on Figure 6.7.

The left figure shows the number of RIDs (indicated by the size of circles) and their correspondence with the complexity metrics (McCabe and maximum nesting). The figure suggests that there is a more direct influence of complexity on fault number than the maximum nesting value. The figure on the right shows the distribution of RIDs according to their severity. Besides other findings, our experiment also showed that there is a clear negative correlation between comment to code ratio and code complexity.

Some high level conclusions on RIDs found in this project:

1. **Comment** RIDS (which are often neglected by the customer, especially if a project is close to mission start) may have been found by ISVV or prevented during development if basic formal methods and automated, "template-like" check were used.
2. In the case of **Minor** and **Major** RIDs, requirement traceability is a key factor in efficient V&V. These result in more objective RIDs which are

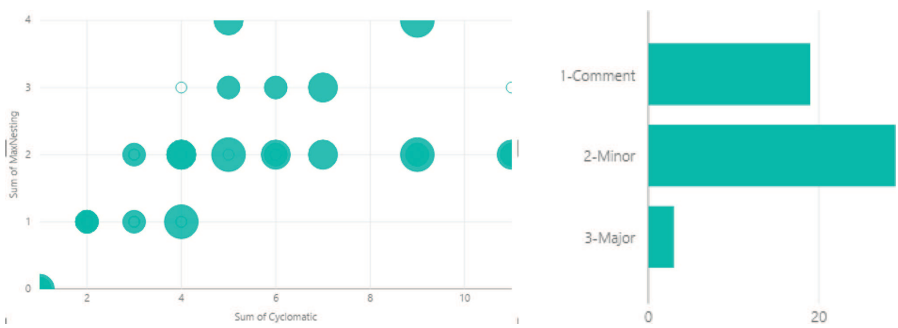

Figure 6.7 Complexity metrics and fault density.

also easier to validate and accept by the customer, while their correction can also be better traced.

Major RIDs may include a wide variety of discrepancies, from potential deadlocks caused by obsolete code to using wrong data types which may result in buffer overflow. Some of these faults are hard to find using manual methods, but are supported by formal tools which also return counter examples representing (potentially) wrong behavior. Major RIDs also require peer review, and are mostly accepted by the customers. Introducing Behavior-Driven Design technologies in the development may also help focusing testing and V&V effort and eliminating faults caused by inconsistent development phases.

6.5 Conclusions

Cost estimation is a fundamental pillar stone of all project management activities. Traditional systems and software engineering can rely on a variety of CEs providing sufficiently accurate predictors on the efforts needed to a particular application development. Independent systems verification and validation of safety critical applications is a crucial activity in assuring the compliance with the standards.

The current chapter evaluated the possibilities of creating a cost estimator dedicated to the V&V phase of system design. The creation of such an estimator is feasible currently primarily due to the unavailability of a sufficiently large calibration dataset. However, a proper adaptation of traditional software CEs has proven its usefulness in process improvements and what-if style evaluation on changes in the workflow.

The adaptation of software cost predictors is not a mechanical process. Differences in assigning a metrics to the complexity of the target project, the scoring of the highly skilled personnel, the impact analysis of the introduction of sophisticated tools, etc. all need a domain expert.

As ISVV is a follow-up activity of a design phase, the design quality heavily influences the amount of work to its execution. The quality of outputs of the previous activities (e.g., design in the V&V Plan, Requirement Verification, Hazard Analysis) are major Effort drivers for rest of the V&V cycle. Modeling this behavior will also allow to perform ROI analyses at early stage of the V&V process and/or continuously monitoring the cost-quality tradeoff of the overall V&V cycle.

A major innovation of the approach is the use of advanced exploratory data analysis techniques [15] to get deep insights into the ISVV process.

Finally, the chapter pinpoints that minimal dataset which is a recommended target of project logging to support future process improvement.

References

[1] CECRIS. *FP7-PEOPLE-IAPP-CECRIS-324334 D2.1 Assessment methodology for analysis of companies V&V processes*.
[2] CECRIS. *FP7-PEOPLE-IAPP-CECRIS-324334 D2.5 Definition of Certification Oriented V&V Processes*.
[3] CENELEC. (1999). *EN 50126: Railway Applications – The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*. Brussels: CENELEC.
[4] CENELEC. (2011). *EN 50128: Railway applications – Communication, signalling and processing systems – Software for railway control and protection systems*. Brussels: CENELEC.
[5] CENELEC. (2003). *EN 50129: Railway applications – Communication, signalling and processing systems – Safety related electronic systems for signaling*. Brussels: CENELEC.
[6] ISO. (2011). *ISO26262 – Road vehicles – functional safety, International Organization for Standardization*.
[7] ISO (International Organisation for Standardisation) and IEC (International Electrotechnical Commission). (2009). "Software Product Quality," in *ISO/IEC 9126*, November 2009.
[8] RTCA. (2012). *RTCA/DO-178C, Software Considerations in Airborne Systems and Equipment Certification*.
[9] Pataricza, A., Gönczy, L., Brancati, F., Moreira, F., Silva, N., Esposito, R., Salánki, Á., Bondavalli A. (2016). "Towards an analysis framework for cost & quality estimation of V&V project," in *DASIA 2016*, Tallin, Estonia.
[10] Brancati, F., Pataricza, A., Silva, N., Hegedüs, Á., Gönczy, L., Bondavalli, A., and Esposito, R. (2015). "Cost Prediction for V&V and Certification Processes," in *2015 IEEE International Conference on Dependable Systems and Networks Workshops (DSN-W)* (New York, NY: IEEE), 57–62.
[11] COCOMO. (2015). *COCOMO II – Constructive Cost Model*. University of Southern California.
[12] Constructive System Engineering Cost Model (COSYSMO). Available at: http://cosysmo.mit.edu/

[13] Fortune, J., Valerdi, R., Boehm, B. W., and Stan Settles, F. (2009). "*Estimating Systems Engineering Reuse*." MITLibraries. Available at: http://dspace.mit.edu/handle/1721.1/84088.

[14] Kan, S. H. (2002). *Metrics and Models in Software Quality Engineering*, 2nd ed. Boston, MA: Addison-Wesley Longman Publishing Co., Inc.

[15] Pataricza A., Kocsis I., Salánki Á., Gönczy L. (2013) "Empirical Assessment of Resilience," in *Software Engineering for Resilient Systems. SERENE 2013*, eds A. Gorbenko, A. Romanovsky, V. Kharchenko. Lecture Notes in Computer Science, vol. 8166. Springer, Berlin, Heidelberg.