Effort estimation model for software development projects based on use case reuse

Katija Rak¹ | Željka Car² | Ignac Lovrek²

¹ IBM Croatia, Croatia
² Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

Abstract
This paper describes a new effort estimation model based on use case reuse, called the use case reusability (UCR), intended for the projects that are reusing artifacts previously developed in past projects with similar scope. Analysis of the widely spread effort estimation techniques for software development projects shows that these techniques were primarily intended for the development of new software solutions. The baseline for the new effort estimation model is the use case points model. The UCR model introduces new classification of use cases based on their reusability, and it includes only those technical and environmental factors that according to the effort estimation experts have significant impact on effort for the target projects. This paper also presents a study which validates the usage of UCR model. The study is conducted within industry and academic environments using industry project teams and postgraduate students as subjects. The analysis results show that UCR model can be applied in different project environments and that according to the observed mean magnitude relative error, it produced very promising effort estimates.

KEYWORDS
effort estimation, reusability, software development, use case, use case points

1 INTRODUCTION

Effort estimation required for a software development project is extremely important for the success of the overall solution delivery. Despite this fact, studies¹ show that a significant progress in improving the performance estimation techniques has not been reported, which represents one of the major challenges within the software industry.² Incorrect effort evaluation often causes budget overrun, delay in delivery, and failure to fulfill contractual obligations and indirectly affects the quality of the product itself. It is therefore not surprising that a very common cause of failure of software development projects in the field of information technology (IT) is incorrect effort estimation.³

The estimated amount of work has a direct impact on several aspects of the software development process life cycle: it may challenge or support a decision on the development of software product depending on the investment justification, it is used as an input parameter for determining the budget of the project and the market price, it affects the project plans, it schedules delivery of project artifacts, etc. In practice, estimated effort is often less than the actual effort at the end of the project.⁴ Such optimistic estimates favor the prospective user (client) because often “optimistic” estimate justifies lower and, therefore, competitively priced products. However, in such circumstances, the same project team in the product development phase meets many challenges—some of which are the most common, already mentioned, exceeding the set budget and delay in delivery.¹⁵ The average deviation or exceeding of the planned amount of work in the research presented in Jorgensen⁵ is about 30%. Reasons for the effort overrun are complex and often were not elaborated in detail in previous research on the effort assessment.¹
On the other hand, an accurate project effort estimate affects the efficient use of existing resources. In cases where the amount of the estimated effort exceeds the amount of the actual effort (the planned amount is overpriced), there is a possibility that human resources are committed to the project to a greater extent than necessary for the execution of the project.

Present trends impose goals such as shorter duration of the software development cycle and cheaper product prices. This directly targets lower project effort, but there is still a parallel demand to maintain the agreed quality of the product. To meet set requirements, software development models introduce the practice of defining software modules and their implementation. Those serve as the core solution with a possibility to reuse certain modules in other (separate) software solutions. Core solution beside program code contains a vast number of development process artifacts, such as requirement descriptions, architecture, use cases, test specifications, etc. Reusability practice of software artifacts improves the productivity and quality of new software products, while reducing the resources, cost, and time of the future software development projects.6-8

Conducted studies include an analysis of the most commonly used effort estimation techniques, and those can be categorized into two groups9-11: algorithmic models based on parameters (constructive cost model, lines of code, functional points, use case points—UCP, etc.) and heuristic approach12 (expert estimation, neural networks, a rule of thumb, techniques, Delphi, etc.).

Analysis of the estimation techniques listed above10,11,12 showed that they are primarily intended for new software development. Due to the lack of an appropriate estimation technique that would consider the reusability aspect, some of the observed cases of development projects from practice showed that the alternative technique used is expert assessment, although with inaccurately estimated effort.

Such results clearly show that there is a strong need for defining a new effort estimate model that would serve the projects which are reusing artifacts developed in previous projects within the same program. General recommendation for project organizations is to automate the estimation procedures9 and adapt tools and approach according to their needs. Tools containing embedded algorithmic models based on parameters that describe the requirements of the software solution provide a standardized assessment process. Therefore, the focus of our research is specifically set on the algorithmic models.

The goal of our research is to define a new effort estimation model intended for projects reusing artifacts developed in previous projects with similar scope. The baseline for a new effort estimation model would be the UCP model, developed by Karner.14 Input factors of the UCP model would be assessed through quantitative research of effort estimation experts and possibly included in the new model. Potentially, new input factors would be defined within this research and incorporated in the new model that would introduce the reusability aspect of previously developed artifacts.

According to our knowledge, so far none of the estimation models described in literature have been dealing with reusability aspect, and our contribution would be a mathematical model for effort estimation which can be adapted for projects that are reusing artifacts previously developed in past projects with a similar scope of work. The term "projects with a similar scope of work" refers to those projects that have similar functional requirements.

The remaining sections of this paper are organized as follows. Section 2 gives an overview of the UCP model which is used as the baseline for a new effort estimation model as part of related work. Section 3 describes our experience and challenges of applying UCP model in projects that are reusing previously developed artifacts. In Section 4, we propose the definition of a new effort estimation model based on use case reuse (UCR). Section 5 describes the mathematical model of UCR and all related algorithms for producing effort estimation, followed by model parameterization in Section 6. Furthermore, Section 7 describes the validation process of the UCR model. In Section 8, application principles of the UCR model are represented, followed by conclusion and future work statements in Section 9.

2 | RELATED WORK

As mentioned before, there is evidently a lack of proven estimation methods considering reusability aspect. Widely used estimation models show good results in estimating effort for software development projects where artifacts are built from scratch. The research is initially focused on the selection of an appropriate existing estimation model and then on its modification with new factors that would describe the reusability aspect.

We searched for the previously performed surveys and systematic literature reviews in the journals and conference proceedings based on keyword topic: software effort estimation. The results found in the literatures1,9,15-18 helped us to assess which of the existing effort estimations for software developments projects would provide the most appropriate baseline for the new effort estimation model.

Based on the surveys and systematic literature reviews, we decided that the most appropriate effort estimation model to be used as the starting one is the UCP, developed by Gustav Karner.14 Within the surveys and systematic literature reviews, UCP was explored in terms of usage and estimation accuracy as (1) original method as defined by Karner, (2) hybrid technique in combination with other estimation technique (eg, constructive cost model), or (3) within broader group of model-based estimates.

The UCP is one of the widely used models to estimate the amount of work to develop software,16,17,19 and, more importantly, this model is suitable for different types of projects, from short duration ones (up to a month), to those over 1 year. There is wide applicability—no dependence on software architecture or programming language (Java, Web Logic, MS Visual Studio, C++, etc.). It was developed primarily for the rational unified process (RUP) development methodology, but it is significant that the UCP is also used in agile software development.16 The next sections will describe the UCP estimation model including the use case concept which forms the basis for effort estimation.

Researches from academia and industry have shown interest in the UCP-based approaches because of the promising results obtained along with their early applicability in budget estimation.19 Some of the methods developed based on UCP approach are iUCP,20 e-UCP,21 and Re-UCP.22
A challenge in adapting an existing model is to understand possible consequences of the adaptation. Terms of usage, mathematical model, and potential limitations must be well understood by the research team. For the UCP model, these elements were clearly described and easily understood by the research team which makes the UCP model an appropriate baseline for building a new effort estimation model.

2.1 Use cases

Use case describes how the user is interacting with the system. It is defined by a list of possible interactions between the system to be developed and external participants (actors) to achieve a specific objective. Actors are people or information systems; one of the actors is the system that is being developed, which also cooperates with other actors. According to the guidelines for writing use cases the most important parts of a use case are as follows: name/title which describes the goal of a certain functionality, actors participating in the use case (people or cooperating systems), use case overview which describes a main activity performed by an actor, main scenario (basic flow) which is the most common sequence of steps leading to the goal, and extensions to the main scenario describing alternative steps associated with the occurrence of some events.

2.2 Use case points (UCP)

The basis of the UCP model lies in the analysis of the system's use cases. The first step is the classification of actors: simple actor is a system that communicates through defined application programming interface, average actor is a system that interacts via TCP/IP protocol, and complex actor represents a participant who interacts through a graphical user interface or web applications. A weight is assigned to each actor category as follows:

- Simple actor, weight: 1,
- Average actor, weight: 2,
- Complex actor, weight: 3.

The total unadjusted actor weight (UAW) is calculated by counting the number of actors in each category, multiplying each total by its specified weight and then adding the products.

The use cases are also classified into simple, average, and complex, depending on the number of transactions in the use case. The transaction marks the event (activity) between the actors and the system. A weight is assigned to each use case category as follows:

- Simple use case (three or fewer transactions), weight: 5,
- Average use case (four to seven transactions), weight: 10,
- Complex use case (more than seven transactions), weight: 15.

The unadjusted use case weights (UUCW) is calculated counting the number of use cases in each category, multiplying each category of use case with its weight and adding the products. The UAW is added to the UUCW to get the unadjusted use case points (UUCP).

Unadjusted use case points (UUCP) are adjusted by the values assigned to technical factors (Table 1a) and environmental factors (Table 1b). These factors affect the amount of work irrespective of the size of the software product, and they describe functional and non-functional requirements, as well as the characteristics of the project team. Each factor is assigned a value between 0 and 5 depending on the assumed impact on the software developed. The value of 0 indicates that this factor is irrelevant to the product, while value 5 means that it is essential.

Multiplying the value of each factor in Table 1a by its weight and then adding all these numbers will get the Tfactor sum. The technical complexity factor (TCF) is calculated as in Equation 1:

\[ TCF = 0.6 + (0.01 \times T_{factor}) \]  (1)

| Table 1A Technical factors in UCP model |
|----------------------------------------|
| Factor      | Description               | Weight | Value  |
|-------------|---------------------------|--------|--------|
| T1          | Distributed system        | 2      |        |
| T2          | Response adjectives       | 2      |        |
| T3          | End-user efficiency       | 1      |        |
| T4          | Complex processing        | 1      |        |
| T5          | Reusable code             | 1      |        |
| T6          | Easy to install           | 0.5    |        |
| T7          | Easy to use               | 0.5    | 0-5    |
| T8          | Portable                  | 2      |        |
| T9          | Easy to change            | 1      |        |
| T10         | Concurrent                | 1      |        |
| T11         | Security features         | 1      |        |
| T12         | Access for third parties  | 1      |        |
| T13         | Special training required | 1      |        |
Multiplying the value of each factor in Table 1b by its weight and adding all the products will get the $E_{\text{factor}}$ sum. The EF is calculated as in Equation 2:

$$EF = 1.4 + (-0.03E_{\text{factor}}).$$  \hspace{1cm} (2)

The adjusted UCP is calculated as follows in Equation 3:

$$UCP = UUCP \times TCF \times EF.$$  \hspace{1cm} (3)

Eventually, in estimating the amount of work required to develop software, it is crucial for a project manager to express the effort in the appropriate unit of measure, eg, man hour (or person hour), man month, etc. For such results, it was necessary to determine how many man hours is needed for one UCP.

Karner suggests the value of 20 man hours per UCP to produce exact effort estimation.\textsuperscript{14} In the other examples from industrial cases suggestions vary from 15 to 30 man hours per UCP\textsuperscript{24}

A different approach is suggested by Schneider and Winters\textsuperscript{25,26} based on the level of experience and stability of the project team and regardless of the software product's technical characteristics. For every EF, the product of weight and assigned value is calculated; the number of factors (Table 1b) from F1 through F6 that have values below 3 are counted and added to the number of factors in F7 through F8 that have values above 3. If the total is 2 or less, 20 man hours per UCP is used; if the total is 3 or 4, 28 man hours per UCP is used. If the number exceeds 4, it is recommended that changes should be made to the project so the number can be adjusted, or alternatively that the number of man hours should be increased to 36 per UCP.

### 3 Case Study—Experience with UCP Model in Industry Projects

Case study was conducted within the software industry program consisting of more than 20 projects which all had a very similar scope of work: development of solutions that enabled the integration of various IT system management applications. Every project covered the scope of integration activities considering specific requirements imposed by an individual client. Applications for system management used by a great number of different clients had either the same or very similar functionalities.

Projects were united into a program to manage them more efficiently due to following guidelines: human resources were shared between the projects in the program, project issues, and risks applicable for two or more projects were managed at the program level, and software development methodology and processes were aligned for all projects.

The program followed software development methodology RUP. Use case models showed that in terms of projects' size, all the projects could be characterized as small sized according to their scope. The number of actors varied from three to four, and number of use cases varied from 10 to 12.

One of the main advantages of uniting projects in a joint program was the sharing of knowledge and lessons learned about the software development process. To achieve maximum utilization of all resources in the program, it was extremely important to manage them properly and efficiently. It had to be considered that the project team consisted of experts with adequate knowledge and skills to perform their assigned role in the project and in accordance with the needs. In addition, it was necessary to make timely decisions about their engagement in the projects. Precise effort estimation for each of the projects was very important for several aspects of the program management; at the program start it was an input for individual project plans, project costs, and resource management.

At the beginning of the program, it was assessed that the solution framework (including all deliverables) should be reusable as much as possible (respecting contractual obligations for each project as well). Reusable artifacts covered not only source code (or parts of source code) but also project documents: scope of work, solution architecture, system context, business process model, use case model, and test specifications. Reusability of specific artifacts was considered during requirements analysis (elaboration phase) and decided during the project build (construction) phase. On the other hand, effort estimation had to be conducted during the planning phase, based on the information known at that phase: high level functional scope and project team characteristics. Project team had to follow carefully very well-defined rules about documenting each of the

| Table 1B | Environmental factors in UCP model |
|---------|-----------------------------------|
| Factor  | Description                       | Weight | Value |
| F1      | Familiar with RUP                 | 1.5    |       |
| F2      | Application experience            | 0.5    |       |
| F3      | Object-oriented experience        | 1      |       |
| F4      | Lead analyst capability           | 0.5    |       |
| F5      | Motivation                        | 1      |       |
| F6      | Requirements stability            | 2      |       |
| F7      | Part-time workers                 | -1     |       |
| F8      | Difficult programming language    | -1     |       |
project deliverable to ensure that other project members could easily comprehend whether such deliverable could be assessed for reuse in other projects. Project deliverables were available for project team in shared document repository (with certain access restriction based on project role). In parallel, project team maintained a tracker of artifacts reusability across projects.

The UCP model was followed to estimate efforts of first five projects and after completion of each project the team compared estimated effort versus actual effort. The estimated effort for the first project where all the project artifacts were developed from scratch was 9% higher than the actual effort. Further in the text the term "initial project" will be used to describe a project where all the project artifacts are developed from scratch (development ab initio). However, for the subsequent projects, where the artifacts from the first projects were reused, estimated efforts according to the UCP model were significantly higher than the actual efforts.

Magnitude of relative error (MRE) as a parameter is important and commonly used criterion in estimation. It is calculated as shown in Equation 4. In fact, MRE is the absolute error in estimating project, and the closer to zero it is, the more accurate is related estimation model.

\[ MRE = \frac{|Actual\ project\ effort - Estimated\ project\ effort|}{Actual\ project\ effort} \]  

(4)

Table 2 shows the MRE of project effort of the first five projects (P1-P5) where the estimated project effort was calculated based on UCP model, as in Equation 4.

The UCP estimation model could provide effort assessment for every individual project, based on the size, functional and non-functional requirements for the particular product developed. Such an approach was appropriate only for the effort estimation of the initial project within the program where all project artifacts were built from scratch. For the subsequent projects, the team could not set any input for the estimation model that would describe that certain solution artifacts were reused from previous projects. As a result, the effort estimated by the UCP model for subsequent projects was not sufficiently accurate as it was significantly higher than the actual effort.

4 | DEFINITION OF EFFORT ESTIMATION MODEL BASED ON USE CASE REUSE (UCR MODEL)

To provide more precise effort estimation for the projects where artifacts are reused, we propose the definition of a new effort estimation model based on use case reuse—called use case reusability (UCR). The UCP model is proved to be efficient for estimating the effort of initial projects so it would be utilized as a starting point for setting the new UCR model. Input factors defined by Karner in the UCP model would be reassessed and potentially included in the UCR. New input factors would also be defined describing reusability aspect. After the input factors and their related attributes (scale of each input factor) are determined, mathematical model including the effort estimation algorithm would be set. The last step of UCR definition process would be appointing of weights to each of the input factor’s attributes.

In the proposed UCR model, use cases would serve as input parameter to compare similarity and compatibility of scope of work between the initial and subsequent projects. Use case comparison between the projects characterize the scale of reused artifacts which does not imply reuse of code only, but potentially other project artifacts such as design model, data models, architectural decisions, and test specifications. Level of reusability of other artifacts increases with the number of equal or similar use cases. Use cases as such are defined in the early stage of the development project, and therefore this estimation model could be used already in the project planning phase.

Our observation after the completion of a few subsequent projects showed three main aspects that determined lower actual effort of subsequent projects comparing to the actual effort of initial project: (1) functional scope of the projects; (2) technical complexity of the solution developed; and (3) environmental factors. Karner based UCP model on the very similar aspects: project scope expressed by UUCP, technical complexity defined by technical complexity factor (TCF), and project team characteristics expressed by environmental complexity factor (EF). Hence, the next step is the assessment of all the factors that made calculation of UUCP, TCF, and EF as defined by Karner and the decision whether they should be included in the new UCR model. To make the assessment more objective, we included in the assessments an expert team that consisted of three project managers and five IT architects, all of them with an experience in effort estimation processes. Project managers were seniors in their roles, all of them with PMP certificate. Senior IT architects had more than 5 years’ experience in their current role, and three of them were IBM Certified Architects. Two project managers and two IT architects had worked in the past on the software development projects that were reusing artifacts from the previous projects that included the reuse of project documents and source code.

### Table 2  Estimated effort (UCP model), actual effort and MRE for projects P1 to P5

| Project Name | Estimated Effort (UCP) in Person Hours | Actual Effort in Person Hours | MRE  |
|--------------|----------------------------------------|------------------------------|------|
| P1           | 1129                                   | 1032                         | 9%   |
| P2           | 740                                    | 283                          | 161% |
| P3           | 649                                    | 207                          | 214% |
| P4           | 649                                    | 197                          | 229% |
| P5           | 649                                    | 195                          | 233% |
All members of the expert team were asked to list additional factors that according to their opinion influence the project effort on top of the factors defined by Karner. All new factors that impacted effort of subsequent projects (either increased or decreased the effort) were discussed between the experts and all of them appertained to three already known main aspects as per Karner: functional scope, technical complexity and environmental factors.

4.1 Functional scope of the projects

Functional scope refers to the services that a specific system offers. Karner decided to describe it in his UCP model by use cases, and the same approach is followed in the UCR effort estimation model. Original classification of use cases and actors by the same criteria into simple, average, and complex is kept in the UCR model as well. However, the UCR model is distinctive by inclusion of additional classification of use cases for the subsequent projects based on their reusable elements:

- New use case (UC_N)—represents completely a new use case that does not represent any part of the previously defined use cases
- Similar use case (UC_S)—represents use case similar to an existing use case from the previous project(s)
- Identical use case (UC_R)—represents an identical use case that had been already defined in the previous project(s)

In the initial project where all project artifacts are built from scratch, all the use cases are defined as new use cases. Each of the subsequent projects has a dedicated functional scope, and each new requirement must be analyzed to determine if it fits the elements of an already existing use case. The requirement analysis is performed in the following way: all the elements of existing use cases (title, description, basic flow, actor, alternative flow, preconditions, and postconditions) have to be read to understand if by content they fit new requirement. Grouping use cases into functional areas helps project team to review smaller number of existing use cases.

Use case is identical (UC_R) to the existing use case only if the requirement is defined by the same elements of an existing use case: they both have the same title, description, basic flow, actor, alternative flow, preconditions, and postconditions.

Use case is similar (UC_S) to the existing use case if the requirement is described only by the same title and description of the existing use case. Other elements such as actors and flows are not identical.

In all the other cases, use case is classified as new one (UC_N).

4.2 Technical complexity

Karner defines 13 technical factors to describe the overall technical complexity of the software product (Table 1a). Scientific studies indicate that the influence of technical factors (TCF) in the UCP model is minor and that these factors do not cover all non-functional requirements of the product. Some of the studies even recommend not to count TCF in UCP calculation (to set the value to 1 in Equation 3). Therefore, the next step is to define appropriate technical factors for the new effort estimation model UCR that would have impact on the calculation of effort for both—initial and subsequent projects.

The task of the expert team is to assess which of the existing technical factors defined by Karner have significant impact on project effort. Significance of Karner's technical factors is evaluated by every member of the team by rating factors as extremely important (awarding 3 points), moderately important (awarding 2 points), or slightly important (awarding 1 point). It is proposed that the new UCR model contains only those factors whose score is at least 80% of the maximum possible total score which is the case for the following factors: Distributed System (T1), Reusable code (T5), and Portable (T8).

Nonetheless, the technical factor Reusable code (T5) is not included in the UCR model as the code as such must be reusable by the nature of the target projects (eg, the code in all projects should be reusable in the subsequent projects). In the same manner, the UCR model does not include factors like flexibility and modularity that describe general reusability aspects that have to be followed in the projects that plan to reuse artifacts across the projects.

The expert team agree to rename factor called Distributed Systems into Software Architecture factor as this term includes the wider characteristics, eg, system distribution, architecture layering, complexity of infrastructure, safety requirements, and non-functional requirements.

According to previous experience, the expert team assesses that an additional factor has a significant impact on the actual effort of subsequent projects and can be classified as a technical factor: integration with new systems. This factor is included in the UCR model, describing if the number of integration points for a software product in the subsequent project is higher than it is the case for the previous project.

4.3 Environmental factors

Environmental factors as defined by Karner are primarily related to the competence of the project team (factors F1-F5). Analysis of the actual amount of five completed projects shows that the experience of team members inversely affects the total effort for subsequent projects.

The task of the expert team is to assess the environmental factors of UCP model in the same way as the technical factors: each member of the team evaluates factors as extremely important (awarding 3 points) or moderately important (awarding 2 points) or slightly important (awarding 1 points).
Again, the new UCR model includes only those factors whose score is at least 80% of the maximum possible total score, which is the case for the following factors: Application experience (F2), Lead Analyst Capability (F4), and Requirements stability (F6). Factor F6 Requirements stability is renamed to Requirements maturity. It covers the quality of given requirements (eg, business process details and business rules).

Considering their previous experience, the expert team introduces the following new factors into UCR model that describe the characteristics of the project team:

- Team colocation—all team members are at the same location or within distributed team (virtual).
- Team size—growing number of team members require more project task integration activities (which can also be placed in the overhead).
- Maintenance of project documentation—availability of quality documentation that describes artifacts of the initial project (the artifacts that are exploited).
- Team cohesion—describes whether the team members and other stakeholders collaborated in previous projects.

The UCR model does not include factors that describe general reusability aspects (eg, flexibility, modularity, and easy to change) that have to be followed in the projects that intend to reuse artifacts across other projects.

All input factors defined for the UCR model are listed in Tables 3a (Technical factors for UCR model) and 3b (Environmental factors for UCR model). Each of the technical and environmental factors has been assigned a descriptive scale of attributes (low, medium, high) with their set of criteria for each attribute. In the UCP model, the estimator must assign a value from 0 to 5 describing the significance of each technical or EF, however with no clear guidance what ambiguously represents the difference between two close values. By setting the clear criteria for each possible attribute of individual factor, input parameters for the estimation model are objective and transparent. Figure 1 depicts the input factors of UCP and UCR models with their attributes. The figure shows the relation of the factors between the estimation models and which of the factors in UCR derived from UCP model.

## 5 | MATHEMATICAL MODEL OF NEW EFFORT ESTIMATION MODEL—UCR

The total unadjusted actor weight in UCR model (UAWR) is calculated as defined by Karner: counting the number of actors in each category, multiplying each total by its specified weight, and then adding the products.

\[
UAW_R = \sum_{i=1}^{n} A_i \text{(complexity)}
\]  

For \(i = 1, 2, \ldots, n\) (\(n\) represents total amount of actors).

Given that the use case is classified in relation to the complexity and reusability, unadjusted use case weight for UCR model (UUCWR) is calculated as the sum of products for each use case weights given for the complexity criteria and reusability criteria.

\[
UUCW_R = \sum_{i=1}^{n} UC_i \text{(complexity)} \times UC_i \text{(reusability)}
\]  

For \(i = 1, 2, \ldots, n\) (\(n\) represents total amount of use cases).

The UAWR is added to the UUCWR to get the UUCP for UCR model (UUPCR).

\[
UUCP_R = UAW_R + UUCW_R
\]  

### TABLE 3A  Technical factors in UCR model

| Index | Technical Factors | Description | Scale of Attributes |
|-------|------------------|-------------|--------------------|
|       |                  |             | Low—L | Medium—M | High—H |
| UCR-T1 | Software architecture | System distribution, multi-tenant architecture, complexity of infrastructure and non-functional requirements | Centralized system, simple infrastructure, a small number of non-functional requirements | Centralized system, moderately complex infrastructure, a moderate number of non-functional requirements | Distributed systems, complex infrastructure, a large number of non-functional requirements |
| UCR-T2 | Portable | Supporting multiple platforms at the same time: Web, mobile applications, various operating systems | Supporting only one platform | Supporting two platforms | Supporting three or more platforms |
| UCR-T3 | Integration with new systems | Integration with larger number of new systems | There is no integration to the other new systems | Integration with one to two systems | Integration with three or more systems |
TABLE 3B Environmental factors in UCR model

| Index | Environmental Factors | Description | Scale of Attributes |
|-------|------------------------|-------------|--------------------|
|       |                        |             | Low—L | Medium—M | High—H |
| UCR-F1 | Application experience | Knowledge of the application by chief architect and lead developer | Chief architect and lead developer do not know the business process of application nor the solution (architecture, requirements, etc.) | Chief architect OR lead developer do not know the business process of application and the solution (architecture, requirements, etc.) | Chief architect and lead developer do know very well the business process of the application and the solution (architecture, requirements, etc.) |
| UCR-F2 | Experience of team members | Previous experience of chief architect and lead developer | Chief architect and lead developer have less than 3 years of experience in their role; poor knowledge of business processes (industry) | Chief architect OR lead developer have more than 3 years of experience in their role; average knowledge of business processes (industry) | Chief architect and lead developer have more than 3 years of experience in their role; knowledge of business processes (industry) is above average |
| UCR-F3 | Maturity of requirements | The quality and stability of the set requirements (necessary details of the business processes, business rules, etc.) | Requirements change frequently, details of business processes are not described. Re-engineering of business processes required | Requirements change rarely, details of business processes are not described. Re-engineering of business processes is not required | Requirements are stable, details of business processes are included and well communicated. Re-engineering of business processes is not required |
| UCR-F4 | Team colocation | Physical distribution of team members | The team members are in different locations and in different time zones | The team members are in different locations and in the same different time zones | The team members are located in the same location |
| UCR-F5 | Team size | Size of team | More than 30 team members | Between 15 and 30 team members | Up to 15 team members |
| UCR-F6 | Maintenance of project documentation | Maintenance of project documentation and artifacts | There is no standardized process for maintaining project documentation | The standard process for maintaining project documentation is defined, but it is not comprehensive or not followed entirely | The standard process for maintaining project documentation and artifacts is defined and followed entirely |
| UCR-F7 | Team cohesion | The degree of the previous collaboration of team members | The project team members have not previously worked together on the project of a similar scope of work | The project team members have previously worked together on the project of a similar scope of work | The project team members have previously worked together on two or more projects of a similar scope of work |

Technical complexity factor in UCR model (TCFR) is counted as product of all technical factors weight, from UCR_T1 till UCR_T3:

$$TCFR = \prod_{i=1}^{3} UCR_{Ti}$$ (8)

For $i = 1, 2, 3$.

Environmental complexity factor in UCR model (EF_R) is counted as product of all EFs weight, from UCR_F1 till UCR_F7:

$$EF_R = \prod_{i=1}^{7} UCR_{Fi}$$ (9)

For $i = 1, 2, 3, ..., 7$.

The premise of the UCR model is that amount of work depends on the attributes of technical and environmental factors. Therefore, factors of technical complexity and environment have been calculated as the products of these factors’ weight.

Adjusted amount of UCR is calculated as the product of UUCW for UCR model (UUCP_R), technical complexity factor (TCFR) and environmental complexity factor (EF_R) - both for UCR model:

$$UCR = UUCP_R \times TCF_R \times EF_R$$ (10)

To reach the final measure of estimated effort in man hours, the next step would be to define the value of man hours per UCR. Karner in his work for UCP model suggested the value of 20 man hours per UCP to estimate the amount of work. 23 For the UCR model, UCR size would be determined through calibration by dividing the actual effort of projects by the size of UCR.

$$UCR\text{[man hours]} = \frac{Actual \ project \ effort \ [man \ hours]}{UCR}$$ (11)

Detailed description of calibration process of UCR measure in man hours is in Section 7.
Historical project data from three different programs (Program A, Program B, and Program C) were collected to (1) calibrate the size of UCR expressed in man hours for UCR model, and (2) to validate the UCR model. Historical project data included the input attributes for the UCR mathematical model, effort estimated by UCP model, and it included the value of actual effort.

Each program had different scope of work and different project teams (resources were not shared within any of the programs). Projects within Program A were industry (commercial) projects where software applications (web/desktop) were developed to integrate various applications. Each of the projects had 10 to 12 use cases with four to six project members.

Projects in Program B were commercial projects where software applications were developed for the clients within public sector. Each of the projects had up to 22 use cases with 8 to 12 project members.

Both programs A and B followed principles of the RUP development methodology so actors and use cases were defined in the Use Case documents.

Program C represented four projects where software applications were developed for persons with complex communication needs, under platform named ICT-AAC (ICT Competence Network for Innovative Services for Persons with Complex Communication Needs), funded by the EU. Each of the projects had from 15 to 22 use cases with four to five project members. The project team followed agile development methodology. Thus, use cases and actors were documented and classified based on application's functionalities by interviewing the project team members.

Project managers of all programs provided information about the attributes for technical and environmental factors and actual project effort expressed in man hours for each project.

Projects and their historical data are divided into two separate sets: calibration set and validation set. A total of 18 projects from Program A make the calibration set and 11 remaining projects from Programs A, B, and C make the validation set. Projects are primarily divided into the sets by the time of their execution. All the projects in calibration set were completed before the projects in the validation set. In both sets there are “initial” and “subsequent” types of projects. Characteristics of each of the Program within calibration and validation set are listed in Table 4.

Next step is to define the factor weights (parameters) for all attributes in the UCR model depending on their impact on overall project effort. The weights of factors that were part of UCP model such as Actors and Use Case complexity are set as in UCP model. The weights of new factors in
UCR model are determined by Delphi method by the five estimation experts that participated in the definition of technical and environmental factors. In Delphi method, a group of experts is led to the agreed opinion of the observed matter. The method is applicable for groups, which helps to avoid the dominant influence of one person which is often present in joint decision making. In the first (preliminary) round, participants give an assessment of each problem individually (usually through questionnaires) and without consultation with the other participants. The evaluation results are collected, processed, and distributed to all participants. In the second round, the participants are asked to re-estimate the answer to the same question, this time with the knowledge of the responses of other participants. The second round usually results in narrowing the range of estimated values and coming up with a joint decision (consensus).

The experts are assessing the impact of environmental factors, technical factors, and use cases classified by reuse (UC_N, UC_S, UC_R) on project effort by estimating their productivity range (PR). PR represents the maximum range of influence that certain factors could have on the workload of the software development for the full range of attribute values (referring to the attribute range from low to high). The value of PR produced by Delphi method was used for the calculation of initial factor weights for each of the attributes.

For example, the 100% value of PR indicates that the individual factor workload (effort) can increase 100% if the factors attribute changes from low to high. For such example a participant would give value 2. On the other hand, for another factor the PR can be 75% which indicates that the amount of work can be increased to 75% if the factors attribute changes from low to high. In such an example the value is 1.75.

Technical and environmental factors are awarded three possible attributes (low, medium, high). The decision process is carried out in two rounds of tests. The experiment results using Delphi method are shown in Table 5 (the second round of tests).

| Index  | Technical factors             | S1 | S2 | S3 | S4 | S5 | Mean value of product. range | Median value of product. range | Mod value of product. range | Range of values |
|--------|-------------------------------|----|----|----|----|----|-------------------------------|------------------------------|-----------------------------|-----------------|
| UCR-T1 | Software architecture         | 1.5| 1.65| 1.45| 1.4| 1.6 | 1.52                          | 1.5                          | #N/A                        | 1.4 - 1.65      |
| UCR-T2 | Portable                      | 1.35| 1.5 | 1.4 | 1.2 | 1.4 | 1.37                          | 1.4                          | 1.2             | 1.2 - 1.5       |
| UCR-T3 | Integration with new systems  | 1.2 | 1.3 | 1.3 | 1.1 | 1.4 | 1.27                          | 1.3                          | 1.15            | 1.15 - 1.4      |

| Index  | Environmental factors         | S1 | S2 | S3 | S4 | S5 | Mean value of product. range | Median value of product. range | Mod value of product. range | Range of values |
|--------|-------------------------------|----|----|----|----|----|-------------------------------|------------------------------|-----------------------------|-----------------|
| UCR-F1 | Application experience        | 1.35| 1.25| 1.2 | 1.15| 1.2 | 1.24                          | 1.25                         | 1.15            | 1.15 - 1.35     |
| UCR-F2 | Experience of team members    | 1.25| 1.25| 1.2 | 1.3 | 1.3 | 1.26                          | 1.25                         | 1.2             | 1.2 - 1.3       |
| UCR-F3 | Maturity of requirements      | 1.15| 1.2 | 1.1 | 1.15| 1.15| 1.15                          | 1.15                         | 1.1             | 1.1 - 1.2       |
| UCR-F4 | Team colocation               | 1.05| 1.05| 1.1 | 1.15| 1.15| 1.1                           | 1.1                          | 1.05            | 1.05 - 1.15     |
| UCR-F5 | Team size                     | 1.05| 1.1 | 1.1 | 1.1 | 1.05| 1.08                          | 1.1                          | 1.1             | 1.05 - 1.11     |
| UCR-F6 | Maintenance of project documentation | 1.05| 1.15| 1.15| 1.1 | 1.15| 1.1                           | 1.1                          | 1.15            | 1.05 - 1.15     |
| UCR-F7 | Team cohesion                 | 1  | 1.1 | 1.05| 1  | 1.05| 1.04                          | 1.05                         | 1               | 1 - 1.1         |

| Index  | Use cases (based on reusability) | S1 | S2 | S3 | S4 | S5 | Mean value of product. range | Median value of product. range | Mod value of product. range | Range of values |
|--------|---------------------------------|----|----|----|----|----|-------------------------------|------------------------------|-----------------------------|-----------------|
| UC_S   | Similar use case                | 0.8| 0.7| 0.7 | 0.8 | 0.7 | 0.74                          | 0.7                          | 0.7             | 0.7 - 0.8       |
| UC_R   | Identical use case              | 0.6| 0.6| 0.55| 0.65| 0.6 | 0.6                           | 0.6                          | 0.6             | 0.55 - 0.65     |
To determine numerical value of each attribute, we need to set ratios of attributes. The values for each of three attributes are defined as linearly proportional. Initial weights are calculated based on the rules described in the following paragraph.

### 6.1 Linear proportions of weights

Weights of attributes for technical and environmental factors are calculated as linearly proportional (L refers to attribute Low, M refers to attribute Medium, H refers to attribute High):

\[
L = x + 1, \quad M = 1, \quad H = 1 - x.
\]

(12)

For technical factors applies that attribute “Low” requires less project effort than attribute “High” (L < H). Therefore, the weight of attribute “Low” is less than the weight of attribute “High” and PR is calculated as follows:

\[
PR_{\text{median}} = \frac{H}{L}.
\]

where \(PR_{\text{median}}\) is the median range value taken from the second (final) round of Delphi method.

Sequentially:

\[
x = \frac{(1 - PR_{\text{median}})}{(1 + PR_{\text{median}})}.
\]

\[
L = \frac{(1 - PR_{\text{median}})}{(1 + PR_{\text{median}})} + 1.
\]

\[
M = 1.
\]

\[
H = 1 - \left(1 - PR_{\text{median}}\right)\left(1 + PR_{\text{median}}\right).
\]

For environmental factors applies that attribute “Low” requires a greater amount of work than attribute “High” (L > H). Therefore, the weight of attribute “Low” is greater than the weight of attribute “High” and PR is calculated as follows:

\[
PR_{\text{median}} = \frac{L}{H}.
\]

where \(PR_{\text{median}}\) is the median range value taken from the second (final) round of Delphi method. Sequentially:

\[
x = \frac{(PR_{\text{median}} - 1)}{(1 + PR_{\text{median}})}.
\]

\[
L = \frac{(PR_{\text{median}} - 1)}{(1 + PR_{\text{median}})} + 1.
\]

\[
M = 1.
\]

\[
H = 1 - \left(PR_{\text{median}} - 1\right)\left(1 + PR_{\text{median}}\right).
\]

Median value of PR for UC_S and UC_R is set as the weight for new classification of use cases – similar (UC_S) and identical use case (UC_R).

### 6.2 Calibration process

The goal of calibration process is to define the size of UCR expressed in man hours for UCR model. For each of the projects in calibration set UCR size is calculated according to the Equation 13:

\[
\frac{\text{UCR (man hours)}(\text{Pi})}{\text{[man hours]}} = \frac{\text{Actual project effort (Pi) [man hours]}}{\text{[UCR(Pi)]}}.
\]

(13)

To derive the size of UCR expressed in man hours, we use historical project data from selected projects within Program A for calibration process of UCR model. We collected historical data from 16 projects that fulfilled the eligible criteria for UCR model application. There were two or more projects with similar scope of work and one or more of them reused the artifacts from the initial project.

UCR value expressed in man hours differs for initial projects once compared with the subsequent projects. Values of UCR for the initial and subsequent project are defined by Equations 14 and 15.
$$UCR_{initial\ project} = \frac{\sum UCR\ (Pi_{initial\ project})}{n(Pi_{initial\ project})}$$  \hspace{1cm} (14)

$$UCR_{subsequent\ project} = \frac{\sum UCR\ (Pi_{subsequent\ project})}{n(Pi_{subsequent\ project})}$$  \hspace{1cm} (15)

Final weights of input factors are listed in Table 6.

7 | VALIDATION OF UCR MODEL

Within the validation process, UCR model is applied on both initial and subsequent projects. Goal of the validation process is to evaluate estimation results with respect to estimation accuracy from the point of view of a project team in the context of industry and academic projects.

7.1 | Experiment planning

Validation of UCR model is performed as an experiment that evaluates the accuracy of estimated project effort when different estimation models are used. The factor in the experiment is the estimation model and the treatments are UCP and UCR models. Subjects are project teams in the industry projects from global IT company (for Programs A and B), and PhD students at Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia (for Program C). Objects for validation are historical project data in the validation set from Programs A, B, and C.

The experts that participated in the definition of factors and weights of UCR model were among project members of certain projects within Program A (they had no relation to Program B nor C). To avoid potential bias in model validation, the project in which those experts have participated are not within validation set of projects; hence, those experts are not within the subjects of experiment.

Accuracy of estimation model is observed by MRE. The MRE is calculated using the Equation 4.

**TABLE 6** | Final weights of input factors in UCR model

| Index  | Actor Description                                      | Weight |
|--------|--------------------------------------------------------|--------|
| A_L    | Application programming interface—API                  | 1      |
| A_M    | System that communicates via TCP/IP                    | 2      |
| A_H    | System that communicates via graphical user interface  | 3      |

| Index                                | Use Case Types (Complexity Classification) | Weight |
|--------------------------------------|-------------------------------------------|--------|
| UC_L                                 | 3 or less transactions                    | 5      |
| UC_M                                 | From 4 to 7 transactions                  | 10     |
| UC_H                                 | From 8 to 15 transactions                 | 15     |
| UC_C                                 | 16 or more transactions                   | 20     |

| Index                                  | Use Case Types (Reusability Classification) | Weight |
|----------------------------------------|--------------------------------------------|--------|
| UC_N                                  | New use case                               | 1.0    |
| UC_S                                  | Similar use case                           | 0.7    |
| UC_R                                  | Identical use case                         | 0.6    |

| Index                                | Technical Factors                         | L      | M      | H      |
|--------------------------------------|-------------------------------------------|--------|--------|--------|
| UCR-T1                               | Software architecture                     | 0.80   | 1.00   | 1.20   |
| UCR-T2                               | Portable                                  | 0.83   | 1.00   | 1.17   |
| UCR-T3                               | Integration with new systems              | 0.87   | 1.00   | 1.13   |

| Index                                | Environmental Factors                     | L      | M      | H      |
|--------------------------------------|-------------------------------------------|--------|--------|--------|
| UCR-F1                               | Application experience                    | 1.11   | 1.00   | 0.89   |
| UCR-F2                               | Experience of team members                | 1.11   | 1.00   | 0.89   |
| UCR-F3                               | Maturity of requirements                  | 1.07   | 1.00   | 0.93   |
| UCR-F4                               | Team colocation                           | 1.05   | 1.00   | 0.95   |
| UCR-F5                               | Team size                                 | 1.05   | 1.00   | 0.95   |
| UCR-F6                               | Maintenance of project documentation      | 1.05   | 1.00   | 0.95   |
| UCR-F7                               | Team cohesion                             | 1.02   | 1.00   | 0.98   |

| Index                                | MANE (size) for initial project           | 10     | Man days |
|--------------------------------------|-------------------------------------------|--------|----------|
| UCR (size) for subsequent project    | 5.5 Man days                             |        |          |
1. Null hypothesis, $H_0$: There is no difference in estimation accuracy (measured by MRE), of estimated project effort between the application of UCP and UCR model.

\[ H_0: \text{MRE (UCP)} = \text{MRE (UCR)}. \]

Alternative hypothesis:

\[ H_1: \text{MRE (UCP)} \neq \text{MRE (UCR)}. \]

The independent variables are project estimated efforts for both estimation models UCP and UCR, and the actual project effort. Dependent variable is MRE. Historical project data contain the values of independent variables to test the hypothesis:

- Estimated efforts as per UCP and UCR models, and actual project effort are expressed in man hours, hence measured on ratio scale.

To test the hypothesis, an experiment with paired design is used as each subject uses both treatments on the same object. Subjects that have participated in Program A have estimated the project effort first with UCP estimation model, while subjects that took part in Program B and C have estimated the project effort first with UCR estimation model. Hence, the balanced design is in place as we have approximately the same number of subjects starting with the first treatment as with the second. The paired design is analyzed with Paired t-test.

Cook and Campbell have defined four types of threats: conclusion, internal, construct, and external validity.

Reliability of measures is considered as a threat to conclusion validity. Use case definition and classification as an input factor in UCP and UCR models require human judgment. Hence, to achieve consistency among initial and subsequent projects, it is necessary to apply consistent rules in use case definition and classification.

Instrumentation is a threat to internal validity. Historical data about actual project effort must be collected and reported in the same way across different programs and different project teams.

Within construct validity, there is a threat to hypothesis guessing. Although the subjects were not part of estimation expert group, some subjects have noticed themselves the trend of effort decrease in subsequent projects.

Relatively small number of available projects for validation of UCR model (small number of initial and sequential projects) is considered as a threat to external validity.

7.2 Experiment execution—data collection

Project historical data were collected in two ways: (1) project managers provided information about actual effort, and (2) project managers (in some cases together with software architects) provided information about estimated project effort per UCP and UCR models. The data were presented to experimenter to ensure that all subjects have reported accurately about historical data and that they applied the UCP and UCR models in correct order. Such approach was utilized to minimize the impact of two threats to validity: reliability of measures and instrumentation.

7.3 Graphical visualization of data

Collected data about project estimated effort (per UCP and UCR models) and actual project effort is shown in Figure 2. Only for one project within validation set (P6) actual and estimated effort (per both models) is significantly higher comparing to others. This data is not considered as outlier.

![Actual and estimated effort of projects](image-url)
and will not be excluded from data set because related project had wider scope then the other projects within validation set. In addition to that, MRE per UCR model for project P6 is below 1%, which is another argument not to treat this data as outliner.

### 7.4 Parametric testing (paired t-test)

Paired t-test is performed as per calculations in Table 7.\textsuperscript{21}

For projects within validation set, MRE project effort estimated by UCP and UCR models are listed in Table 8. Null hypothesis states that there is no difference in estimation accuracy (measured by MRE) of estimated project effort between the application of UCP and UCR model. The alternative hypothesis states that there is a difference.

The results of paired samples test are presented in Table 9.

The number of degrees of freedom is $f = n - 1 = 11 - 1 = 10$, $t_0 = 4.686$ and $t_{0.025,10} = 2.228.31$. Since $t_0 > t_{0.025,10}$ it is possible to reject null hypothesis at the 0.05 level.

Limited number of projects (samples) within validation set can represent a threat to external validity. However, according to the results of paired t-test, it is possible to reject null hypothesis.

### 7.5 Results of the experiment

By rejecting null hypothesis, experiment shows there is a difference in estimation accuracy (measured by MRE), of estimated project effort between the application of UCP and UCR models.

The two-tailed $P$ value equals 0.0009. By conventional criteria, this difference is considered statistically significant.

Accuracy of estimation model is observed as well by these criteria: MMRE (mean magnitude of relative error) and PRED (Percentage of Predictions within 20%) of estimated project effort (20).

#### TABLE 7  Calculation of paired t-test

| Item                  | Description input                                                     |
|-----------------------|-----------------------------------------------------------------------|
| Input                 | Paired samples: $(x_1, y_1), (x_2, y_2) ... (x_n, y_n).               |
| $H_0$                 | $\mu_d = 0$, where $d_i = x_i - y_i$, the expected mean of the differences is 0 |
| Calculations          | $t_0 = \frac{\bar{d}}{s_d / \sqrt{n}}$, where $s_d = \sqrt{\frac{\sum(d_i - \bar{d})^2}{n-1}}$ |
| Criterion             | Two sided ($H_1: \mu_d \neq 0$): Reject $H_0$ if $|t_0| > t_{\frac{\alpha}{2}, n-1}$. Here, $t_{\frac{\alpha}{2}, f}$ is the upper $\alpha$ percentage point of the $t$ distribution with $f$ degrees of freedom. One sided ($H_1: \mu_d > 0$): Reject $H_0$ if $|t_0| > t_{\alpha, n-1}$.

#### TABLE 8  MRE of estimated efforts for UCP and UCR model for validation set of projects

| Validation Set of Project | P1  | P2   | P3   | P4   | P5  | P6   | P7   | P8   | P9   | P10  | P11  |
|---------------------------|-----|------|------|------|-----|------|------|------|------|------|------|
| MRE (UCP)                 | 125.5% | 307.2% | 365.4% | 266.7% | 306.9% | 12.7% | 76.9% | 77.3% | 372.9% | 56.6% | 204.1% |
| MRE (UCR)                 | 15.8%  | 1.2%  | 13.6% | 2.4%  | 9.0% | 0.5% | 27.3% | 6.6% | 3.9% | 5.1% | 0.6% |

| MMRE (UCR)               | 7.81% |
| MdMRE (UCR)              | 5.07% |
| PRED(20) (UCR)           | 91%   |

#### TABLE 9  Validation of UCR model—paired t-test

| Paired Samples Test | Mean       | Std. deviation | Std. error mean | 95% confidence interval of the difference | $t$ | df | Sg. (2-tailed) |
|---------------------|------------|----------------|-----------------|------------------------------------------|-----|----|----------------|
| UCP MRE-UCR MRE     | 1.896678   | 1.342326       | .404727         | .994891 - 2.798465                      | 4.686 | 10 | .001           |
All MRE values are used to calculate the value of MMRE following the equation:

\[
MMRE = \frac{1}{n} \sum_{i=1}^{n} MRE.
\] (16)

The MMRE is calculated to indicate the relative amount by which the estimated effort is an underestimate or overestimate in comparison to the actual effort. MMRE is used in most of the research work as evaluation criterion due to its independent-of-units characteristic which means that MMRE is independent of units of estimated effort like man hours, man months, etc. MMRE is a meaningful tool used to summarize statistics and is very important in evaluating a software effort estimation model.\(^\text{19}\)

\(PRED(x)\) is the percentage of estimates that are within \(x\%\) of the actual efforts. \(PRED(20)\) is the percentage of estimated effort that is within 20\% of the actual efforts.

Both measures, MMRE and \(PRED(20)\), show promising results once calculated for projects within validation set for UCR model (included in Table 8).

### 8 | APPLICATION OF THE UCR MODEL

The initial prerequisite for performing an effort estimation using the UCR model is to form a team of experts with proper knowledge of the effort estimation process and experience in projects that are reusing artifacts. On the other hand, it cannot be expected that there is a great number of such experts within an organization.

For UCR model, the estimators need to set all input factors for every project (initial and subsequent projects):

- Number of actors and their classification by complexity criteria
- Number of use cases and their classification by complexity and reusability criteria
- Technical factors concerning the software product
- Environmental factors concerning the project team

If required, the expert team might adapt UCR model for their specific environment by performing the following steps:

1) Assessment of input factors in proposed UCR model by the expert team (review of all aspects that influence the project effort required in their project environment)
2) Potential addition or removal of input factors
3) Adjustment of \(TCFR\) or \(EFR\) equations (Equation 8 and 9) if input factors have been changed
4) Setting the weights of attributes for new input factors by conducting Delphi experiment
5) Calibration of UCR size expressed by man hours

There are no constraints in usage of UCR model in projects with different reuse approaches (eg. patterns, domain engineering and Software Product Line). In such cases, the above listed steps can also be applied to customize the UCR model to specific project environment.

Another potential scenario of applying UCR model is to perform only the last step from the adaptation process described above: calibration of UCR size (expressed in man hours) for the certain project team. UCR size indicates the productivity of a certain project team\(^\text{33}\) and based on the historical project data of actual effort and input factors for other projects within that organization, different value of UCR size (for both initial and subsequent projects) might be applicable for another project team.

### 9 | CONCLUSIONS AND FUTURE WORK

This paper introduces a new effort estimation model, the UCR, intended for software development projects that are reusing previously developed project artifacts. UCR model is modification of the existing UCP effort estimation model developed by Karner\(^\text{14}\) with the elements that are describing the reusability aspect.

The UCR model application shows improvement of estimated effort results compared with the UCP model for the observed projects within the case study (absolute values of MRE and MMRE for UCR model are lower than absolute values of MRE and MMRE for UCP model, and \(PRED(20)\) for UCR is higher than \(PRED(20)\) for UCP model). In practice, more precise effort estimation allows the project team to improve overall project management process—planning, scheduling, resource management, and cost estimation. Projects whose historic data was used for model calibration and validation are dominantly small sized in terms of project scope (small or moderate number of deliverables to be produced) and team size (up to five team members), so certain deviations in effort estimation might be expected for medium or large projects. Therefore, in the future
research, the focus would be on wider UCR model validation in medium or large projects—for both initial and subsequent projects to increase acceptability.

As part of future work, there is a possibility for medium or large sized projects to enlarge the scale of factor attributes in the UCR model (eg, by introducing very low and very high attributes). Project size and project complexity are increasing from small to large sized projects: medium and large sized projects have a moderate to high number of deliverables which are usually technically more complex, number of team members rises and timeframe for delivery expands. Additional granularity of factor attributes would allow more precise differentiation between the projects in terms of technical complexity solution and project team characteristics.

In medium or large projects, a greater number of use cases is expected. Use case classification in terms of reusability would be time-consuming if their comparison was done manually. Therefore, as part of future work there is an opportunity to introduce the process of automated comparison of use cases potentially as part of a newly developed tool.

This paper has focused on the reuse of artifacts across projects with similar scope. The UCR model was validated separately in three different programs that had different scope and context. As part of future work, authors will consider exploring reuse across projects with different scope. Such approach will have to be studied thoroughly in terms of the impact on project effort. Level of reuse across projects with different scope is expected to be lower than for the projects with similar scope. At the other hand, artifacts reused from the projects with different scope might need adaptation, which very likely will require additional project effort.

**ORCID**

Katija Rak [http://orcid.org/0000-0003-1262-9538](http://orcid.org/0000-0003-1262-9538)

**REFERENCES**

1. Molakka, K., Jorgensen M. A review of software surveys on software effort estimation. International Symposium on Empirical Software Engineering. Rome, Italy: Simula Res. Lab. Lysaker Norway; 2003

2. Grimstad, S. Understanding of estimation accuracy in software development projects. 11th IEEE International Software Metrics Symposium (METRICS 2005)

3. Jorgensen M, Grimstad S. The impact of irrelevant and misleading information on software development effort estimates: a randomized controlled field experiment. IEEE Trans Softw Eng. 2011;37(5):695-707. [https://doi.org/10.1109/TSE.2010.78](https://doi.org/10.1109/TSE.2010.78)

4. Jorgensen M, Grimstad S. Software development estimation biases: the role of interdependence. IEEE Trans Softw Eng. 2012;38(3):677-693. [https://doi.org/10.1109/TSE.2011.40](https://doi.org/10.1109/TSE.2011.40)

5. Jorgensen M. What we do and don't know about software development effort estimation. IEEE Trans Softw Eng. 2014;31(2):37-40. [https://doi.org/10.1109/MS.2014.49](https://doi.org/10.1109/MS.2014.49)

6. Clements P, Northrop L. Software Product Lines: Practices and Patterns. Addison-Wesley Longman Publishing Co., Inc; 2001.

7. Linden FJ v d, Schmid K, Rommes E. Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer-Verlag New York, Inc; 2007.

8. Fazal-e-Amin, Ahmad Kamil Mahmood, Oxley Alan: A proposed reusability attribute model for aspect oriented software product line components

9. Nasir, M: A survey of software estimation techniques and project planning practices, software engineering, artificial intelligence, networking, and parallel/distributed computing, 2006. SNPD 2006. Seventh ACIS International Conference on Year: 2006 Pages: 305-310, [https://doi.org/10.1109/SNPD-SAWN.2006.11](https://doi.org/10.1109/SNPD-SAWN.2006.11)

10. Agarwal R, Kumar t M, Yogesh SM, Bharadwaj RM, Anantzart Infosys Technologies Limited D, Calcutta I. Estimating software projects. ACM SIGSOFT Softw Eng Notes. 2001;26(4):60-67.

11. Boehm Barry, Abts Chris, Chulani Sunita University of Southern California, Los Angeles, USA, IBM Research, 650 Harry Road, San Jose, CA, USA. "Software development cost estimation approaches—a survey". Ann Softw Eng 10, 1-4 (2000) 177–205, Year of publication: 2000 ISSN: 1022-7091

12. Milicic, D.; Wohlin, C.: Distribution patterns of effort estimations, Euromicro Conference, 2004. Proceedings. 30th, Pages: 422-429, DOI: [https://doi.org/10.1109/EURMIC.2004.1333398](https://doi.org/10.1109/EURMIC.2004.1333398)

13. Kusumoto, S.; Matukawa, F.; Inoue, K.; Hanabusa, S.; Maegawa, Y.: Estimating effort by use case points: method, tool and case study, Software Metrics, 2004. Proceedings. 10th International Symposium, Pages: 292-299; 2004, DOI: [https://doi.org/10.1109/METRIC.2004.1357913](https://doi.org/10.1109/METRIC.2004.1357913)

14. Karner, G. Metrics for objectory. Diploma thesis, University of Linköping, Sweden. No. LITH-IDA-Ex-9344:21; 1993.

15. Rastogi, H.; Dhandkar, S.; Kakkar, M.; A survey on software effort estimation techniques, Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference, Pages: 826-830, DOI: [https://doi.org/10.1109/CONFLUENCE.2014.6949367](https://doi.org/10.1109/CONFLUENCE.2014.6949367)

16. Usman Muhammad, Mendes Emilia, Weidt Francila, Britto Ricardo: Effort Estimation in Agile Software Development: A Systematic Literature Review, PROMISE '14, September 2014, Torino, Italy

17. Ribu K., Estimating object-oriented software projects with use cases; 2008.

18. Anda Bente, Dreiem Hege, Jorgensen Magne, Sjøberg Dag: Estimating software development effort based on use cases—experience from industry. In M. Gogolla, C. Kobryn (Eds.): UML 2001—The Unified Modeling Language. Springer-Verlag, 4th International Conference, Toronto, Canada, October 1-5, 2001, LNCS 218: 2001.

19. Kirmani MM, Wahid A. Impact of modification made in re-UCP on software effort estimation. J Softw Eng Appl. 2015;8(06):276-289.

20. Nunes, N. J., iUCP—estimating interaction design projects with enhanced use case points. Proceedings of the 8th International Conference on Task Models and Diagrams for User Interface Design, Springer, 131–145; 2010.

21. Periyasamy, K. & Ghode, A., Cost estimation using extended use case point (e-UCP) model, International Conference on Computational Intelligence and Software Engineering; 2009
22. Kirmani MM, Wahid A. Revised use case point (re-UCP) model for software effort estimation. Int J Adv Comp Sci Appl(IJASCA). 2015;6(3):65-71.
23. Cockburn A. Writing Effective Use Cases. Addison-Wesley; 2000.
24. Sparks S, Kaspczynski K. The Art of Sizing Projects. Sun World.
25. Schneider and Winters: Applying Use Cases. Addison-Wesley; 1998.
26. Stutzke R.D. Estimating Software-Intensive Systems: Projects, Products, and Processes; Addison-Wesley; 2005.
27. Mohagheghi, P., Anda, B., Conradi, R.: Effort estimation of use cases for incremental large-scale software development, Proceeding ICSE '05 Proceedings of the 27th international conference on Software engineering, Pages 303-311
28. Kemerer CF. An empirical validation of software cost estimation models. CACM. 1987;30(5):416-429.
29. ICT competence network for innovative services for persons with complex communication needs, http://www.ict-aac.hr/index.php/en/projects/ict-aac (February 15, 2018)
30. Babic Jurica, Slivar Ivan, Car Zeljka, Podobnik Vedran: Prototype-driven software development process for augmentative and alternative communication applications, 2015 13th International Conference on Telecommunications (ConTEL), 2015, 1-8
31. Wholin C et al. Experimentation in Software Engineering. Springer-Verlag; 2012.
32. Cook TD, Campbell DT. Quasi-Experimentation—Design and Analysis Issues for Field Settings. Boston: Houghton Mifflin Company; 1979.
33. Alves Luis M, Sousa André, Ribeiro Pedro, Machado Ricardo J: An empirical study on the estimation of software development effort with use case points, 2013 IEEE Frontiers in Education Conference (FIE), 2013, 101-107

How to cite this article: Rak K, Car Ž, Lovrek I. Effort estimation model for software development projects based on use case reuse. J Softw Evol Proc. 2019;31:e2119. https://doi.org/10.1002/smr.2119