Empirical Software Engineering (2024) 29:153
https://doi.org/10.1007/s10664-024-10508-x

Forecasting software indicators: an industry-academia collaboration

Claudia Ayala[1] · **Cristina Gómez**[1] · **Martí Manzano** · **Antonin Abherve**[2] · **Xavier Franch**[1]

Accepted: 27 May 2024 / Published online: 23 September 2024

Abstract

Context Nowadays software-development organizations are urged to exploit their data for empowering their decision-making processes. Such data may be used to monitor the status of meaningful software indicators (e.g., software quality, productivity and on-time delivery) that are relevant for their decision-making processes. Forecasting the values of such indicators may provide evidence of a potentially high risk or opportunity that could help to anticipate actions accordingly. Most of the existing forecasting proposals in software engineering use open-source data rather than data from industrial projects. Therefore, there is a lack of evidence on how these proposals fit the particular needs of a software-development organization and how they can be automated into the organization's infrastructure.

Objective To enable software indicators´ forecasting in a software-development organization (Modeliosoft).

Method We designed an industry-academia collaboration based on Action Design Research (ADR) to address Modeliosoft's forecasting challenges.

Results A tool-supported method called FOSI (Forecasting Of Software Indicators) for enabling forecasting in Modeliosoft. We obtained positive results regarding its suitability and technical feasibility in a pilot project of the organization. In addition, we provide details and reflections on the potential usefulness of the method for addressing similar field problems.

Conclusions The procedures and results detailed in this paper are valuable to: 1) address Modeliosoft's forecasting challenges 2) inspire other software-development organizations on how to deal with similar problems and even reuse some procedures and software support tools resulted from this work, 3) promote the win-win benefits of industry-academia collaborations.

Keywords Software-development organizations · Forecasting · Software indicators · Metrics

Communicated by: Markus Borg

Extended author information available on the last page of the article

1 Introduction

Nowadays, the use of data analytics is becoming a crucial asset for helping organizations to improve their managerial operations and decision making, make better use of resources and leverage competitive advantage (Bosch 2016; Marshall et al. 2015). This also includes software-development organizations, which are defined as organizations that can be private (i.e., software companies) or public, extensively developing software either for third-parties, or for internal use (Manzano et al. 2021). Software-development organizations produce large amounts of data related to their software processes and products from the use of their corporate tools (e.g., continuous inspection tools, continuous integration tools, project management tools and issue trackers) that may be exploited to improve their operations and decision making (Martínez-Fernández et al. 2018).

In this context, the software engineering literature has proposed a plethora of techniques based on data analytics to support software-development organizations' decision-making processes (Tosun et al. 2020; Figalist et al. 2021; Misirli and Bener 2014). Most of the existing proposals focus on computing the values of diverse software indicators from the actual evidence gathered in available repositories. Software indicators refer to measures providing an estimate/value that a software-development organization considers important for its decision-making processes. For instance, Vasilescu et al. (Vasilescu et al. 2015) and Choetkiertikul et al. (Choetkiertikul et al. 2018) define and estimate specific software indicators such as *software quality* and *productivity*, and *delivery capability* respectively. Other proposals, instead of focusing on specific software indicators, generalize the definition and estimation processes by supporting the computation of any meaningful software indicator (e.g., (Manzano et al. 2021; Mendes et al. 2018)). The main potential benefit of automating the estimation of such software indicators is the possibility of monitoring their values, thus providing valuable insights to diverse software-development organization's roles for making better informed decisions (Mendes et al. 2018).

However, as data mining and data analytics techniques get mature, software-development organizations are demanding advanced decision-making capabilities not only for monitoring their software indicators but also for estimating the future values of such indicators (Taylor and Letham 2018), i.e., forecasting software indicators. Forecasting is the process of making predictions of the future, mostly based on past and present data, and trends (Chambers et al. 1971). Monitoring the forecasted values of meaningful software indicators may provide funded evidence of a potentially high risk or opportunity that could help diverse roles of the software-development organization to anticipate actions accordingly (i.e., to prevent undesired effects). For instance, forecasting the *Product Readiness* indicator of a software product under development might reveal a likely risk of deadline violations, therefore decision makers can promote earlier the allocation of additional resources to avoid such undesired effect.

Numerous proposals have emerged in the realm of forecasting software indicators, addressing aspects such as reliability (Amin et al. 2013; Kumaresan and Ganeshkumar 2020), software vulnerabilities (Roumani et al. 2015; Yasasin et al. 2020), and software defects (Raja et al. 2009; Wu et al. 2010; Thorström 2017)). However, a significant majority of these proposals often emphasize showcasing the efficacy of algorithms or suggested processes, rather than assessing their feasibility within organizational contexts and their existing infrastructure (Eken et al. 2021; Baltes and Ralph 2022). This dearth of industrial evidence raises concerns about the potential integration of existing proposals into software-development organizations' infrastructure (Eken et al. 2021). As a result, there is a pressing need for research that not only demonstrates the effectiveness of forecasting algorithms but also provides insights into their practical implementation and automation within software-development organizations.

The work presented in this paper is part of an ongoing long-term industry-academia collaboration with Modeliosoft, a private European software-development organization (hereafter software-development company or company for short). The overall objective of this collaboration is to enhance Modeliosoft's decision-making processes through short-term industry-academia collaboration cycles. This paper specifically delves into the cycle dedicated to addressing Modeliosoft's forecasting challenges. Such challenges were approached under Action Design Research (ADR) premises (Sein et al. 2011), that allowed the joint collaboration of the organization (Modeliosoft) with the researchers, for envisaging a Modeliosoft's forecasting solution whose design and evaluation was conducted within the organization. In accordance with ADR premises, the major contributions of the work reported here are:

a) The design of a method called FOSI (**F**orecasting **Of** **S**oftware **I**ndicators) for enabling forecasting in Modeliosoft. This method incorporates state-of-the-art forecasting approaches, and it is complemented with some software support tools to facilitate its application.
b) Preliminary evaluation of the FOSI method in a pilot project within Modeliosoft. This was tackled as a case study, followed by a feedback evaluation from Modeliosoft's practitioners that participated in it. The obtained results show promising outcomes, demonstrating the suitability and feasibility of the method and its support tools within Modeliosoft.
c) Formulation of four design principles derived from the experience of designing and applying the FOSI method in Modeliosoft. These principles offer succinct insights into how the FOSI method addresses the forecasting challenges of Modeliosoft. They help to facilitate the adoption of the procedures and tools proposed by the FOSI method by other organizations facing similar problems.

In addition, our results might promote the win-win benefits of industry-academia collaborations. At this respect, Song and Runeson (Song and Runeson 2023), and Mikkonen et al. (Mikkonen et al. 2018) claim that reporting the results from industry-academia collaborations is especially important for promoting the relevance and transference of software engineering research. Other researchers also support the need of disseminating this type of collaborations in order to foster the impact of research into the industrial practice (Basili et al. 2018; Devanbu et al. 2018).

The content of the paper is organized as follows. Section 2 provides background on ADR and details the previous collaboration with Modeliosoft. Section 3 provides an overview of the research approach followed to deal with Modeliosoft's forecasting challenges. Sections 4–6 detail the ADR-based stages aimed to design and evaluate the FOSI method for Modeliosoft, as well as leverage the knowledge gained from such design and evaluation. Section 7 tackles limitations and threats to validity. Section 8 discusses the lessons learned from this industry-academia collaboration. Finally, Section 9 summarizes the conclusions and future work.

2 Background

2.1 A brief on Action Design Research

As highlighted by Wohlin and Runeson (Wohlin and Runeson 2021), selecting the right research methodology is crucial for industry-academia collaborations to maximize their benefits and ensure success. Typical research methodologies for developing and evaluating solutions to address an industrial challenge in empirical SE are: action research (Elden and Chisholm 1993; Avison et al. 1999), design science (Wieringa 2014), and technology transfer models (Mikkonen et al. 2018). These methodologies are typically characterized by their aim to bring about a change in practice (action research), produce an artifact (design science), or transfer knowledge about a specific practice (technology transfer models). However, despite their respective strengths, each methodology also presents certain limitations, as noted by Cronholm and Göbel (Cronholm and Göbel 2019), and Wohlin and Runeson (Wohlin and Runeson 2021). Action research, for instance, excels in fostering intervention and knowledge emergence within authentic settings, yet it may not sufficiently prioritize innovation and abstraction. Similarly, design science emphasizes abstraction and innovation in artifact creation but often defers evaluation to a subsequent stage, thereby potentially hindering timely feedback and refinement. Moreover, technology transfer models, while effective in promoting collaboration, may sometimes prioritize the collaborative aspect over problem-solving efficacy, necessitating the involvement of multidisciplinary initiatives to overcome inherent limitations. In the realm of Information Systems, another methodology pertinent to industry-academia collaboration has emerged: Action Design Research (ADR) (Sein et al. 2011; Cronholm and Göbel 2022). There is evidence on the successful use of ADR across various fields (Haj-Bolouri et al. 2018; Cronholm and Göbel 2019), and it is gaining attention in empirical software engineering, as underscored by Ralph et al. (Ralph 2014) and a recent publication by Díaz et al. (Díaz et al. 2022). The integration of ADR into empirical software engineering methodologies offers promising avenues for enhancing collaboration effectiveness and advancing research outcomes.

ADR is a research methodology that has demonstrated that it is possible to apply action research to solve an organization's problem together with design science to produce an artifact that solves that problem (Sein et al. 2011). It is, the organizational problem is tackled through an artifact (i.e., a new method or a new tool) whose design and evaluation is conducted within the organization (Sein et al. 2011).

ADR conceives artifact design as a result of a researcher-practitioner collaboration within an organization. Figure 1 reproduces ADR stages and principles (Sein et al. 2011). The first three stages conform an iterative sequence where tasks are intertwingled till distilled into the final learnings at the end of the project. Arrows stand for influential flows.

The description of each ADR stage is as follows:

Problem Formulation. The first stage is triggered by a problem or need encountered in practice or predicted by researchers. It serves as a catalyst for developing a research strategy. This stage draws on two principles: *Practice-Inspired Research* and *Theory-Ingrained Artifact (*Sein et al. 2011*)*. The former emphasizes viewing organization problems as knowledge-creation opportunities. The second principle highlights that the intervention (i.e., the IT artifact) is to be informed by theories and existing knowledge that grounds design decisions.

Building, Intervention, and Evaluation (BIE). This stage builds upon the problem framing and theoretical premises adopted in stage one. These premises provide a platform for generating the initial design of the IT artifact. From here on, the IT artifact is further shaped by organizational use and subsequent design iterations using Sein et al's principles (Sein et al. 2011): *Reciprocal shaping* (i.e., the IT artifact and the organization feedback each other: prototypes or pilot projects serve to profile the interpretation of the organizational environment that help a better fit in subsequent versions), *Mutually influential roles* (i.e., researchers and practitioners bring complementary insights),

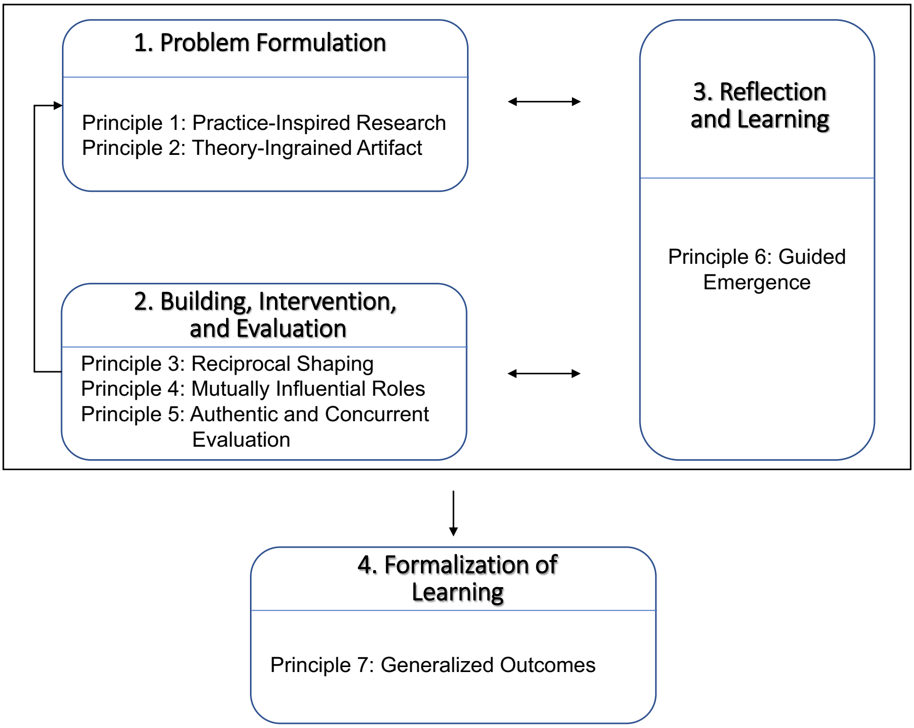

Fig. 1 The ADR method (from (Sein et al. 2011))

and *Authentic and concurrent evaluation* (i.e., authenticity is a more crucial element for ADR than controlled conditions, thus assessment should take place within the company and throughout the research).

Reflection and Learning. ADR involves more than merely solving a problem to an organization. To guarantee that contributions to knowledge are made, conscious reflection on the problem framing, theories adopted, and the emerging IT artifact are critical. The principle is termed as *Guided emergence* where 'emergence' captures this notion of unanticipated consequences that arise during the intervention in the organization and to which researchers should be sensitive to (Sein et al. 2011).

Formalization of Learning. At this point, we reach an artifact that brings with it some premises about the problem framing and the organization setting (i.e., an *ensemble artifact*). Both, the problem and the artifact can be generalized. Sein et al. (Sein et al. 2011) suggests three levels for achieving such generalization: (1) generalization of the problem instance, (2) generalization of the solution instance, and (3) generalization of mechanisms through design principles. Design principles transcend the specifics of IT implementation, focusing instead on the abstract mechanisms that underpin the utility and functionality of a solution. They should provide cues about the goal that need to be achieved and how, so enabling the understanding of how to use the solution.

2.2 Previous Collaboration with Modeliosoft

The research presented in this paper has been performed in the context of a long-term industry-academy collaboration between Softeam Group[1] and the GESSI research group at the Polytechnical University of Catalunya[2]. Softeam is a software-development organization with more than 1300 employees, providing “high-quality services and solutions in strategy, consulting, finance, digital, big data, analytics, performance and operations”. Our collaboration started in 2017 in the context of Q-Rapids (H2020 EU project - Quality Aware Rapid Software Development project n.d.), a European-funded research project. We interacted specifically with Modeliosoft, one of the Softeam’s subsidiary firms. The general goal of the collaboration is to find feasible solutions for improving Modeliosoft’s decision-making processes. As suggested by the continuous and collaborative technology transfer model proposed by Mikkonen et al. (Mikkonen et al. 2018), the vision of our long-term collaboration is based on promoting business case driven short-term goals (mainly promoted by Modeliosoft’s needs) that can be rapidly validated in short collaboration cycles, leading to prompt results. Thus, the initial collaboration cycle resulted in the development and implementation of a method called SESSI (Specification and Estimation of Software Strategic Indicators), which has been published elsewhere (Manzano et al. 2021). The SESSI method enabled Modeliosoft to specify, assess, and monitor relevant software indicators by leveraging corporate repositories and expert knowledge. Since then, some software indicators have been continuously monitored in the company through their corporate dashboard.

This section describes the Indicator Estimation Models used in Modeliosoft and their monitoring infrastructure in order to understand the context where the forecasting solution proposed in this paper is envisaged.

The Indicator Estimation Models used in Modeliosoft are based on Bayesian Networks (BN). A BN is a probabilistic graphical model that represents a set of nodes (variables) and their conditional dependencies via a directed acyclic graph (DAG) (Pearl 1985). BNs are a well-known instrument to build models from data and/or domain experts’ knowledge for the estimation of variables under uncertainty conditions (Tosun et al. 2017). The construction of these models in Modeliosoft relies on domain experts’ knowledge and is supported with corporate repositories’ data exploitation as detailed in (Manzano et al. 2021).

The meaning of a software indicator is represented through a hierarchical structure of nodes. Such hierarchy is considered the DAG of the BN. The top-most-level node of the hierarchy represents the software indicator itself. At its turn, the software indicator is hierarchically decomposed into other meaningful aggregations called derived measure nodes (i.e., intermediate nodes). The lowest-level nodes are called base measure nodes. In addition, each node has associated a Conditional Probability Table (CPT). CPTs enclose information provided by domain experts, such as: a) the potential states of the node, and b) the conditional probability distribution of the node given the different states of its parents (if any).

As an example, Table 1 shows the hierarchical definition of the *Product Readiness* indicator (currently used in Modeliosoft), the semantic of each node and its potential states as defined by domain experts from the company. It also denotes the data sources where the values of the base measures are extracted. Figure 2 illustrates the *Product Readiness* Estimation Model as a BN and its inferred values for a specific point of time.

[1] www.softeamgroup.fr/en

[2] https://gessi.upc.edu/en

Table 1 Hierarchical definition and semantic meaning of *Product Readiness*

Name	Type of Node	States	Description
Product Readiness	Software Indicator	NotReady, Neutral, AlmostReady, Ready	Degree of fulfillment of the features planned for the release and without critical bugs.
Activities Completion	Derived Measure	VeryLow, Low, Medium, High, VeryHigh	Represents the status of the completion of activities planned for the release, including development and specification tasks.
Development Task Completion	Base Measure (extracted from Open Project)	VeryLow, Low, Medium, High, VeryHigh	Represents the fulfillment of the required specification tasks for the release under development.
Specification Task Completion	Base Measure (extracted from Open Project)	VeryLow, Low, Medium, High, VeryHigh	Represents the fulfillment status of the required development tasks for the release under development.
Known Remaining Defects (Closed) Ratio	Derived Measure	Low, Medium, High	Measures defects/ bugs/crashes that lie outside the major bug category and can be deferred to next releases.
Postponed Issues (Closed) Ratio	Base Measure (extracted from Mantis)	Low, Medium, High	Ratio of the minor severity closed issues with respect to the total number of low severity issues.
Product Stability	Derived Measure	VeryLow, Low, Medium, High, VeryHigh	Measures the status of the operational software quality of the monitored release, considering the presence of major issues and the testing status.
Build Stability	Base Measure (extracted from Jenkins)	VeryLow, Low, Medium, High, VeryHigh	Percentage of successful builds with respect to the total of builds triggered in a seven-day period.
Passed Tests Percentage	Base Measure (extracted from Jenkins)	VeryLow, Low, Medium, High, VeryHigh	Percentage of tests passed with respect to the total number of tests executed for the latest build.
Critical Issues (Closed) Ratio	Base Measure (extracted from Jenkins)	VeryLow, Low, Medium, High, VeryHigh	Ratio of high severity closed issues (of type Crash, Block, Major) with respect to the total number of high severity issues.

The inference process to compute the nodes' states and their probabilities is performed through belief propagation over the BN. It requires that base measure nodes are connected to corporate repositories for collecting base measure values (through specially designed data collectors). Base measure values serve to feed base measure nodes and propagate such evidence up to the BN.

Enabling this inference process requires that the continuous values of each base measure extracted from corporate repositories are discretized. Such discretization is mainly required by domain experts as they find that quantifying the information of each CPT using discrete states is easier than using continuous values. In addition, the definition of discrete states allows domain experts to denote semantics into the labels of the states at their convenience (e.g., bad-good, low-high, not ready-ready). Therefore, when building the Indicator Estimation Model, domain experts have to define discretization functions to transform the numerical values collected from data collectors into the corresponding states of each base measure node.

For example, from Fig. 2, we can observe that when the base measure values for *Development Task Completion* and *Specification Task Completion* are collected and discretized into the "VeryHigh" state and the rest of the base measure values (*Postponed Issues Closed Ratio, Build Stability, Passed Tests Percentage, and Critical Issues Closed Ratio*) are discretized into the "Medium" state. These states will propagate up and infer the probability distribution $\{NotReady = 13.2\%, Neutral = 25.9\%, AlmostReady = 38.2\%, Ready = 22.7\%\}$ for the *Product Readiness* indicator. Note that the *Product Readiness* Estimation Model does not provide deterministic assessments but actual probabilities. Given the combination of base measure values fed, the estimation model provides the *AlmostReady* state as the most probable one for the software indicator in such specific time. This can be interpreted as the presence of uncovered critical issues and untested code may hinder the readiness of the product to a great extent.

Some of the benefits of using the previous BN-based structure for the Indicator Estimation Model as stated in (Manzano et al. 2021) are:

a) The hierarchical structure of the Indicator Estimation Model allows explainability as the states of the software indicator can be traced down in the hierarchy, thus to the causes that led the states of each node.
b) The probabilistic nature of the Indicator Estimation Model helps to deal with the uncertainty of typical situations in software development. For example, the fact that software repositories might have some values missing (Twala et al. 2005; Mockus 2008; Twala and Cartwright 2010) (i.e., periods where there are no registered base measure values). So, in cases where some values are missing, the Indicator Estimation Model uses the prior probabilities of the base measure nodes for inferring the states of the software indicator and its derived measures.

To enable the monitoring infrastructure, the Indicator Estimation Model and its corresponding data collectors were operationalized using Netica® software and the APIs provided by the *unBBayes* BN framework (Matsumoto et al. 2011), respectively. For the case of the *Product Readiness* indicator, the inferences are computed daily. Therefore, its associated data collectors were configured to run daily for collecting data and computing the base measure values that feed the *Product Readiness* Estimation Model. Further details of the architecture of the monitoring infrastructure used in Modeliosoft are provided in (Manzano et al. 2021).

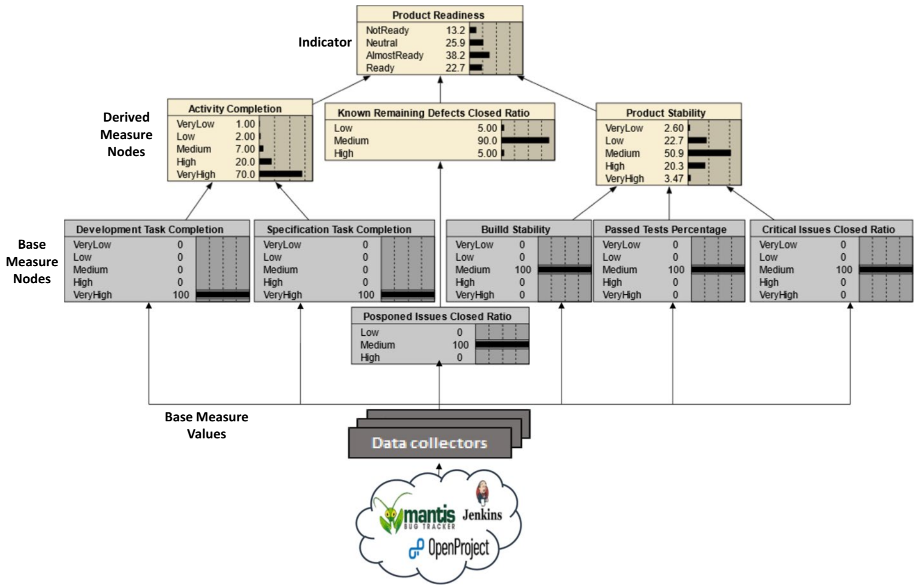

Fig. 2 *Product Readiness* Estimation model

3 Research Design

The work presented in this paper refers to a new collaboration cycle with Modeliosoft. It builds upon the previous cycle and was motivated from the company's desire to enhance their decision-making processes by incorporating forecasting capabilities. The focus of this collaboration cycle was on envisaging a forecasting solution for Modeliosoft and the evaluation of its suitability and feasibility. So, the general research question was:

How to envisage a suitable and feasible forecasting solution for Modeliosoft?

Various factors and considerations guided our methodological decisions regarding how to approach this industry-academia collaboration. Alongside the fundamental assumption of establishing short-term collaboration cycles, some of the most relevant ones include:

- *The intended forecasting solution should be packaged as an artifact.*
  - Modeliosoft's intention was to package the forecasting solution as an "organizational procedure or method", essentially an ensemble artifact that encompasses both procedures and software support tools. This would facilitate the transfer of acquired knowledge to other projects within the company.
  - The researchers agreed on the benefits of packaging the forecasting solution, recognizing its potential for dissemination. In addition, the researchers emphasized the significance of shaping the resulting artifact to the organizational context, as highlighted by Wohlin et al. (2015).

- *Deep researchers' involvement.*
  - Modeliosoft acknowledged a lack of expertise in forecasting solutions, which demanded a deep researchers' involvement to provide the necessary support.
  - The researchers were sought to get deeply involved in the problem solution in order to enhance their capacity to co-produce knowledge in industrial contexts and transfer such knowledge to other similar contexts.

Taking these factors into account, we opted to employ Action Design Research (ADR) (Sein et al. 2011). As detailed in Section 2.1, ADR entails collaborative endeavors between researchers and practitioners to devise artifacts, such as new methods or tools, with the aim of addressing specific organizational issues. Notably, ADR underscores that artifact design and evaluation take place within the same organizational context. This was especially helpful for our purpose of envisaging a short-term collaboration cycle. Moreover, ADR facilitates the generation of prescriptive design knowledge applicable to similar challenges across diverse contexts, aligning with the researchers' interests of transferring knowledge to other contexts. Therefore, we deemed ADR as a suitable choice to fulfill the collaboration requirements and foster a mutually beneficial outcome for both industry and academia.

An ADR Team was formed with a combination of researchers (the UPC authors who had previously participated in the earlier collaboration cycle) and two representatives from Modeliosoft. One of the representatives held a high-level strategic position within the company, while the other had a project leader role with extensive operational knowledge.

Utilizing the stages and principles outlined by ADR (as detailed in Section 2.1), the ADR Team came to a consensus on the corresponding goals, research questions and most suitable activities for each ADR stage under the constrains and resources of our collaboration.

Therefore, in alignment with the overarching research question of our collaboration cycle, each stage of the ADR-based process was designed to fulfill a specific goal corresponding to one or more research questions, as delineated below. Table 2 provides a summary of these goals, research questions, activities, and the main assets generated.

- Problem Formulation:

- RQ 1.1- *What are the forecasting challenges faced by Modeliosoft and which class of problems do they exemplify?* This inquiry was addressed through activities 1 and 2 (refer to Table 2), with detailed discussions presented in Sections 4.1 and 4.2 respectively.
- RQ 1.2- *How can we envision a forecasting solution for Modeliosoft based on the current state of the art?* Activities 3 and 4 (see Table 2) laid the groundwork for tackling this question, with further elaboration provided in Sections 4.3 and 4.4 respectively. It is worth noting that these activities set the basis for the subsequent ADR stage (Building, Intervention, and Evaluation).

- Building, Intervention and Evaluation (BIE):

- RQ2.1- *How to build Indicator Forecasting Models that help to automatically forecast the values of the software indicators currently used in Modeliosoft?* This question prompted an iteration by the ADR team resulting in the design of the FOSI method, outlined as activity 1 of the BIE stage in Table 2, and detailed in Section 5.1.
- RQ2.2- *How is the use of the FOSI method in the pilot project?* This query aimed to guide the design and evaluation of the FOSI method in a Modeliosoft pilot project, as documented in Table 2 and further expounded upon in Section 5.2.

Table 2 Details of activities, goals and assets' production

Stage	Goal	Main Activities	Main Produced Assets	ADR Principle applied
Problem Formulation	Identify and scope the Modeliosoft's forecasting problem as well as the class of problems it exemplifies.	1.- Understand Modeliosoft's forecasting problem and needs. (RQ1.1) - Discussion sessions. - Semi-structured interviews. - Review of existing procedures, and infrastructure (databases and tools in the company).	Set of requirements that should be addressed by the intended solution. (See Section 4.1)	*Practice-inspired research* (P1): Research was driven by a vision to deal with the forecasting challenges of Modeliosoft by co-creating a forecasting solution that serves as a knowledge creation opportunity for other organizations that face similar problems.
		2.- Cast the problem as an instance of a class of problems. (RQ1.1) - Discussion sessions.	Assessment of Modeliosoft problem wrt the problems it exemplifies. (See Section 4.2)	
		3.- Review the literature on indicator forecasting approaches in SE. (RQ1.2)	Related Work on software indicators forecasting approaches. (See Section 4.3)	*Theory-Ingrained Artifact* (P2): The artifact should be informed by existing theories. After a literature search, we envisaged a preliminary forecasting solution based on existing forecasting methods.
		4.- Envisage a preliminary forecasting solution based on the existing literature. (RQ1.2)	Preliminary Forecasting solution. (See Section 4.4)	

Table 2 (continued)

Stage	Goal	Main Activities	Main Produced Assets	ADR Principle applied
Building, Intervention and Evaluation (BIE)	Iteratively design and evaluate a forecasting solution for Modeliosoft.	1.- Design of the forecasting solution. (RQ2.1) - Brainstorming sessions. - Hands-on sessions using datasets from Modeliosoft to develop procedures and software support tools for forecasting.	Modeliosoft's Forecasting solution: FOSI Method. (See Section 5.1)	*Reciprocal shaping* (P3): the design of the FOSI method was influenced by Modeliosoft and the researchers. *Mutually Influential Roles* (P4): The co-creation process of the ADR Team provided complementary insights.
		2.- Evaluation of the FOSI method. (RQ2.2) - Design and execution of a case study to apply the FOSI method in a Modeliosoft's pilot project. - Explicit feedback evaluation from Modeliosoft practitioners.	Evaluation of the FOSI Method in a Pilot Project within Modeliosoft. (See Section 5.2)	*Authentic and Concurrent Evaluation* (P5): The assessment of the FOSI method was done in an authentic Modeliosoft setting.
Reflection and Learning	Reflect on the Modeliosoft problem and its solution, evaluating adherences to ADR principles and interventions results	1.- Reflection on the design of the solution during the whole ADR project. (RQ3) 2.- Evaluate adherences to ADR principles. (RQ3) 3.- Analyze intervention results according to stated goals to adjust accordingly. (RQ3)	Insights from all discussion's sessions among the ADR Team throughout the project. (See Section 4 and 5)	*Guided Emergence* (P6): The ADR Team co-created the FOSI method under the ADR principles and facing organizational and technological challenges from Modeliosoft.
Formalization of Learning	Abstract the learnings for helping other organizations	1.-Discussion sessions and decisions taken during previous stages. (RQ4)	-Design principles to apply the FOSI method. (See Section 6)	*Generalized Outcomes* (P7): To abstract and generalize as much knowledge gained as possible.

- Reflection and Learning:

– RQ3- *How can the ADR-Guided Emergency principle be effectively applied throughout the collaboration cycle?* As indicated in Table 2, this inquiry was addressed through the execution of three activities that spanned the entire collaboration cycle. These activities served as guiding principles for our decisions and adjustments made throughout the collaboration cycle and are reported within the context of the other stages.

- Formalization of Learning:

– RQ4- *How can the insights gained from the design and evaluation of the FOSI method be abstracted to assist other organizations facing similar challenges?* To distil the insights gleaned from the design and evaluation of the forecasting solution for Modeliosoft, we performed the activity outlined in Table 2, as elaborated upon in Section 6.

Figure 3 illustrates the participants and outputs of each ADR stage. Table 2 and Fig. 3 also assist the reader by indicating the section that provides further details on each aspect.

4 ADR: Problem Formulation

4.1 Understanding Modeliosoft's Forecasting Problem and Needs

To identify and scope the organizational problem, the ADR Team engaged in a thorough analysis of the current practices and forecasting challenges within Modeliosoft. To do so, the ADR Team interacted in situ with key stakeholders involved in decision-making tasks (2 leaders with strategic responsibilities, 2 project leaders and 2 operational personnel) through semi-structured interviews and focus groups. The goal was to collect data and information related to their forecasting needs, decision-making processes, software indicators in use and data availability. The execution of the semi-structured interviews and focus groups is reported in (Q-Rapids, Q-Rapids Deliverable D3.1 2018). As a result, an initial set of high-level forecasting requirements were stated, as shown in Table 3.

4.2 Casting Modeliosoft's Forecasting Problem as an Instance of a Class of Problems

Forecasting requirements can vary widely depending on the context and the specific needs of the organization or project. To cast the Modeliosoft's forecasting problem as an instance of forecasting problems that face similar organizations, we integrated the activities of eliciting and assessing the Modeliosoft's forecasting problem with literature review and discussion sessions with colleagues from other projects related to decision-making support, specifically, with colleagues that participate in Q-Rapids (H2020 EU project - Quality Aware Rapid Software Development project n.d.), VISDOM[3] and D3M[4]. These colleagues had first-hand knowledge about usual requirements regarding forecasting in the organizations

[3] Visual Software Diagnostics project. https://visdom-project.github.io/website/

[4] Automated Data-Driven Decision Making. D3M project. https://d3m.upc.edu/en

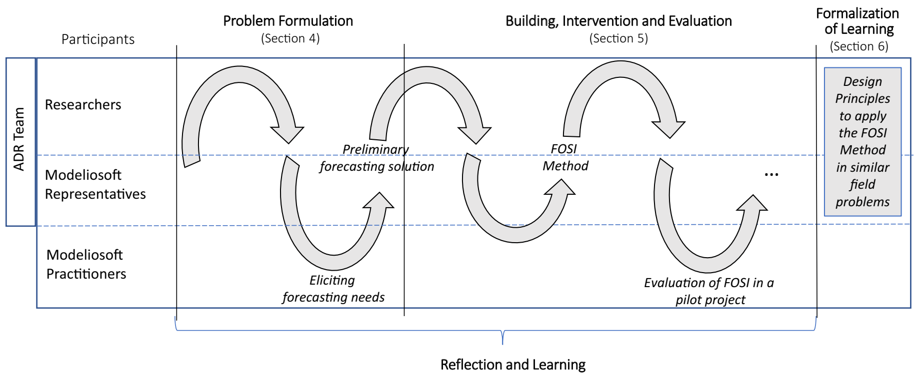

Fig. 3 ADR stages involved in the development of the FOSI method

participating in these projects. This holistic approach enabled us to conceptualize the problem as a representative instance of forecasting problems within software development organizations. Through these collaborative efforts, the ADR team distilled that R1, R2, and R3 are reasonably shared requirements among software-development organizations, while R4 and R5 exhibit a higher degree of dependency on each organization's unique infrastructure.

4.3 Related Work on Software Indicators Forecasting Approaches

In addressing Modeliosoft's forecasting needs, we searched for existing proposals within the domain of software indicators forecasting. In this section, we detail our assessment of the coverage of Modeliosoft's requirements by such existing proposals and discuss their limitations. A summary is provided in Table 4.

Despite the lack of comprehensive solutions meeting all of Modeliosoft's requirements, these existing proposals still provided valuable insights and ideas that help us to shape the tailored solution for Modeliosoft.

Regarding **R1** (*Automated software indicators forecasting*), we considered two key aspects:

- **Indicator adaptability**: This aspect aims to evaluate the proposal's capability to forecast the values of diverse software indicators rather than being limited to specific ones. This is because Modeliosoft's R1 requirement emphasized the need for forecasting models capable of predicting values for any software indicator. Our analysis (see Table 4) highlights a prevailing pattern in existing literature, wherein the majority of works focus on developing forecasting models tailored to specific software indicators (e.g., reliability, software defects, maintainability, work effort, quality, technical debt, etc.), rather than offering a generalized approach applicable across various indicators. Notably, Thorström's work (Thorström 2017) stands out as an exception to this, emphasizing the potential for more versatile methodologies. This observation underscores a significant limitation within current proposals, particularly for companies like Modeliosoft, which rely on multiple software indicators.

Table 3 Modeliosoft's requirements for the forecasting solution

Requirements	Modeliosoft's Requirements	Description from Modeliosoft
Automated software indicators forecasting	R1	To have an automated solution for helping several roles of the company (e.g., CEO, project leaders) to better inform their decisions based on the forecasting values of software indicators.
Missing values and explainability	R2	The solution must deal with partial missing values of corporate repositories and provide explainability about the forecasted values of the software indicators.
Minimal specialized knowledge to build forecasting models	R3	To minimize the specialized knowledge required for putting forward a forecasting solution. It is because most employees do not have extensive forecasting knowledge. The idea is that the solution is adapted as much as possible to the technical background of project leaders of the company, so that they are able to put forward the solution in their corresponding projects with minimal support.
Deployment into the organizational infrastructure	R4	To reuse as much as possible current Modeliosoft's processes, data, and software tools (i.e., infrastructure).
	R5	To use the current Modeliosoft's dashboard for visualizing the forecasted values of the software indicators.

- **Corporate Repositories.** This criterion evaluates the proposal's ability to leverage corporate knowledge stored in repositories such as continuous inspection/integration tools, project management platforms, or issue trackers. While all the reviewed proposals used data repositories, the majority relied on open-source or public datasets such as PROMISE[5] or ISBSG[6] datasets. This indicates a predominant focus on demonstrating algorithmic efficacy rather than assessing feasibility within real organizational contexts and existing infrastructures, as such datasets lack a genuine organizational context that allows organizational deployment.

The intended Modeliosoft's forecasting solution should be designed to provide a flexible forecasting approach capable of forecasting the values of any software indicator from corporate knowledge repositories, thereby enabling Modeliosoft to gain valuable insights from their corporate knowledge.

In regard to **R2** (*Missing values and explainability*), we explored how existing forecasting proposals in software engineering dealt with the following aspects:

- **Missing values:** This aspect evaluates the proposals' ability to explicitly address how to handle missing values in the datasets. A notable observation is that a significant portion of existing forecasting proposals (12 out of 17) do not provide explicit guidance on handling missing data periods. This omission poses a considerable constraint, especially considering that software companies often contend with large volumes of heterogeneous longitudinal data, frequently interrupted by roll-backs or the incremental nature of the software development process. Addressing this constraint is paramount for the development of forecasting methodologies capable of effectively navigating the complexities inherent in corporate repositories.
- **Explainability:** This criterion denotes the capacity of the proposal to provide explainability for the forecasting values. Despite the increasing demand for explainability capabilities to enhance decision support systems (Chazette and Schneider 2020), most existing forecasting proposals overlook the incorporation of explainability features. Nevertheless, some of these proposals utilize indicators comprising aggregated lower-level metrics (e.g., (Wagner 2009; Van Koten and Gray 2006)) or integrate causal models into their forecasting processes (e.g., (Amin et al. 2013; Kumaresan and Ganeshkumar 2020)). These elements present promising avenues for enhancing the explainability of forecasted software indicators values. We got inspired from such aggregation of lower-level metrics presented in these existing works when designed the preliminary forecasting solution for Modeliosoft.

Our investigation revealed that while machine learning and deep learning techniques have garnered attention and achieved relevant results across different domains, their lack of interpretability (Rique et al. 2023) rendered them unsuitable for the forecasting purposes of Modeliosoft. Conversely, Bayesian networks provide explainability of the knowledge encapsulated within their representations and are also capable to deal with missing values (Tosun et al. 2017; Rique et al. 2023). This influenced our decision of basing the Modeliosoft's forecasting solution on Bayesian network premises (as explained below).

For **R3** (*Minimal specialized knowledge to build forecasting models*), we examined various key aspects, including the proposals' prescriptiveness, provision of tool support,

[5] http://promise.site.uottawa.ca/SERepository/datasets-page.html
[6] https://www.isbsg.org/

Table 4 Assessment of existing software indicator's forecasting proposals

Reference	R1 *Automated software indicators forecasting*		R2 *Missing values and explainability*		R3 *Minimal specialized knowledge to build forecasting models*			R4- R5 *Deployment into the organizational infrastructure*
	Indicator Adaptability	Corporate Repositories	Missing values	Explainability	Prescriptive steps	Tool support	Forecasting Approach	Industrial deployment
Amin et al. (Amin et al. 2013)	✗	✗	✗	✗	✓	✗	SM	✗
Kumaresa et al. (Kumaresan and Ganeshkumar 2020)	✗	✗	✗	✗	✓	✗	SM	✗
Roumani et al. (Roumani et al. 2015)	✗	✗	✗	✗	✓	✗	BM	✗
Yasasin et al. (Yasasin et al. 2020)	✗	✗	±	✗	✗	✗	C	✗
Raja et al. (Raja et al. 2009)	✗	✗	✓	✗	✓	✗	SM	✗
Eken et al. (Eken et al. 2021)	✗	✓	✗	✗	✓	✓	BM	✓
Staron et al. (Staron et al. 2010)	✗	✓	✗	✗	✓	✗	SM	✓
Wu et al. (Wu et al. 2010)	✗	✗	✗	✗	✓	✗	SM	✗
Thorström (Thorström 2017)	✓	✓	✗	✗	✓	✗	BM	✓
Wagner (Wagner 2009)	✗	✗	✗	✗	✓	±	SM	✗
Van Koten et al. (Van Koten and Gray 2006)	✗	✗	✗	✗	✗	±	SM	✗
Chug et al. (Chug and Malhotra 2016)	✗	✗	±	✗	✗	±	C	✗
Reddivari et al. (Reddivari and Raman 2019)	✗	✗	✓	✗	✗	✗	C	✗
Priya et al. (Priya Varshini and Kumari 2020)	✗	✗	✗	✗	✗	✗	C	✗
Pospieszny et al. (Pospieszny et al. 2018)	✗	✗	✓	✗	✓	✗	EM	✗
Tsoukala+2020 (Tsoukalas et al. 2020)	✗	✓	✗	✗	✓	✓	BM	✓
Ardimento et al. (Ardimento et al. 2022)	✗	✗	✗	✗	✓	✗	SM	✗
Modeliosoft's Forecasting solution	✓	✓	✓	✓	✓	✓	EM	✓

✓ covered ✗ not covered/not mentioned ± partially covered

Model Comparison (C), Single Model (SM), Best Fitting Model (BFT), Ensemble Model (EM) and their forecasting characteristics facilitating the construction of forecasting models. By assessing these aspects, our aim was to analyze whether the proposals offer a user-friendly approach to building forecasting models, thereby reducing the necessity for specialized expertise and improving accessibility for a wider range of users within organizations like Modeliosoft.

- **Prescriptive steps** for model construction: This aspect assesses whether the proposal offers detailed guidance on the necessary steps for constructing the forecasting model. It serves as an indicator of the level of specialized knowledge required for implementation. As observed in Table 4, various proposals offer, to varying degrees, informal prescriptions of the steps involved in building the forecasting models.
- **Tool support:** This criterion evaluates whether the proposal integrates tool support to facilitate the construction of the forecasting model. The presence of tool support can notably diminish the requirement for specialized expertise and streamline the modeling process. Despite several proposals providing prescriptive steps for building forecasting models, the availability of tool support was relatively limited.
- **Forecasting approach:** This aspect aims to delineate the scope of the proposed forecasting and how the forecasting model is constructed. Our analysis, shown in Table 4, reveals that the assessed forecasting proposals can be categorized as:
  - Comparison (C): These proposals aim to compare the efficacy of different forecasting techniques (e.g., ARIMA, Decision Trees, BN, Logistic regression) rather than to propose a specific forecasting model for a task at hand.
  - Single Model (SM): These works aim to build a forecasting model based on a single forecasting technique.
  - Best Fitting Model (BM): These proposals employ multiple forecasting techniques and selects the best model for the task at hand.
  - Ensembled Model (EM). Ensemble modeling involves creating multiple diverse models to obtain intermediate outcomes, which are combined through particular mechanisms, to predict the forecasted outcome. For example, in the work of Pospieszny et al. (Pospieszny et al. 2018), an ensemble averaging of three machine learning algorithms are proposed for forecasting software project effort and duration.

The diversity of forecasting approaches discovered led us to realize that proposals categorized as (C) or (SM) did not align with Modeliosoft's goal of a forecasting solution suitable for any software indicator. Comparisons (C) were disregarded because they do not propose any forecasting model but instead focus on comparing forecasting techniques. Single Models (SM) were discarded because a single forecasting technique is not suitable for pursuing forecasting for any software indicator (Chambers et al. 1971).

The literature has emphasized that Ensembled Models boosts prediction accuracy (Pospieszny et al. 2018), therefore, for devising the Modeliosoft's forecasting solution we decided to use an Ensembled Model approach together with Best Fitting Model principles to find suitable forecasting models.

R4-R5 (*Deployment into the organizational infrastructure*) cater to specific infrastructure requirements customized for Modeliosoft's needs. Our objective was to investigate whether other proposals offer insights into deploying forecasting models in industrial settings, thereby enabling us to extract valuable lessons. In this context, we documented whether existing forecasting proposals provide comprehensive details on **Industrial Deployment**.

Our analysis, as outlined in Table 4, highlights a significant gap in the literature regarding industrial deployment within forecasting proposals. Only 4 out of 17 proposals provided some form of industrial evidence, albeit without further infrastructure details.

Consistent with our findings above, it is evident that very few works utilize data from corporate repositories; instead, they rely on open-source repositories or public datasets (Eken et al. 2021; Baltes and Ralph 2022). This reliance on external data sources hampers these proposals' ability to access crucial organizational context and infrastructure, thereby hindering their effectiveness in addressing industrial deployment challenges.

4.4 Envisaging a Preliminary Forecasting Solution for Modeliosoft

Based on the insights gained in the previous activities, the ADR Team reached a consensus on conceptualizing a preliminary solution: to build an ensembled model, called Indicator Forecasting Model based on the Indicator Estimation Model (already existing in Modeliosoft and inspired on Bayesian network premises, as described in Section 2.2). The principles of Best Fitting Model would be applied to build forecasting models based on state-of-the-art forecasting techniques for each base measure node of the Indicator Estimation Model that requires it. The decision to pursue this preliminary solution was based on the following reasons:

- In-house expertise: Modeliosoft possessed internal knowledge and expertise in building Indicator Estimation Models. This aligns with R3 and R4, which emphasized leveraging in-house know-how and capabilities.
- Availability of historical data: The existing Indicator Estimation Models had been previously used, which ensured the availability of historical data on estimations of some software indicators. This availability of data, along with the presence of data collectors, met R4 and provided a valuable resource for developing the forecasting model.
- Handling missing values and explainability: The current BN-based nature of the Indicator Estimation Models already addressed potential issues related to missing values and explainability (Manzano et al. 2021). This alignment with R2 was crucial, as incomplete data and explainability are significant challenges faced by state-of-the-art forecasting techniques in real-world applications (Schelter et al. 2018). By building upon the existing structure, the complexity of finding a suitable forecasting solution was reduced.
- Automation, Reusability and Integration: the use of the Indicator Estimation Models as the basis for assembling the Indicator Forecasting Model facilitates the reusability and integration of new components (R1) into Modeliosoft's dashboard for visualizing the forecasting results. This met R5 and ensured a seamless integration of the forecasting capabilities within Modeliosoft's existing infrastructure.
- Cost-effectiveness: Implementing the preliminary solution based on the existing models (R4) was considered the most cost-effective alternative. It minimized the need for extensive retraining, significant changes to the infrastructure, or the acquisition of new technologies or tools.

5 ADR: Building, Intervention and Evaluation

Based on the reasons outlined in the previous section, the ADR Team decided to proceed with assembling the forecasting solution.

To do so, the ADR Team collaborated closely to conceive the Indicator Forecasting Model, guided by the principles of *Reciprocal Shaping* (P3) and *Mutually Influential Roles* (P4). This collaborative approach ensured that such construction was informed by the insights and expertise of both Modeliosoft representatives and the researchers, who shared a background in software development. Modeliosoft representatives contributed domain expert knowledge and knowledge about the company's decision-making expectations, and provided access to proprietary data and infrastructure. Researchers brought forecasting experience and business processes expertise to guide the team in defining suitable processes and developing software support tools for supporting the construction of the Indicator Forecasting Model.

The ADR team leveraged anonymized data obtained from Modeliosoft's repositories. A portion of this data was utilized to prototype procedures for constructing the Indicator Forecasting Model, while the remainder was employed to validate the effectiveness of the resulting model. This iterative process involved hands-on experimentation with the data and testing to devise a tool-supported approach for constructing an Indicator Forecasting Model exploiting the available data and domain expert's knowledge from Modeliosoft. Subsequently, the ADR team focused on deploying the resulting Indicator Forecasting Model onto the company's actual infrastructure. This iterative approach ultimately culminated in the development of a method called FOSI (Forecasting Of Software Indicators) for Modeliosoft.

5.1 Modeliosoft's Forecasting Solution: The FOSI Method

The FOSI (Forecasting Of Software Indicators) method comprises two distinct phases designed to establish a robust infrastructure for forecasting software indicators. Phase 1 provides the steps for constructing and evaluating an Indicator Forecasting Model leveraging domain experts' knowledge and repository data from the Modeliosoft's repositories. It addresses the requirements R1, R2, and R3. Once the Indicator Forecasting Model meets the performance threshold deemed acceptable by domain experts, Phase 2 involves its deployment within Modeliosoft's infrastructure. Such deployment addresses requirements R4 and R5. Figure 4 provides an overview of the FOSI method, illustrating the sequential flow of activities across both phases.

The details of each phase of the FOSI method are presented below.

5.1.1 Phase 1: Building and Evaluating an Indicator Forecasting Model

Based on the BN nature of the Indicator Estimation Model, its corresponding base measure nodes are fed with the discretized forecasted values of each base measure. The aim is to propagate such states up to the BN and get the forecasted states and probabilities for all nodes up to the software indicator.

To forecast the values of the base measures, additional models/databases should be built using the available historical data. This phase of the method has five steps. The first two steps require information that should be defined by a role that understands why and for what the intended Indicator Forecasting Model is needed.

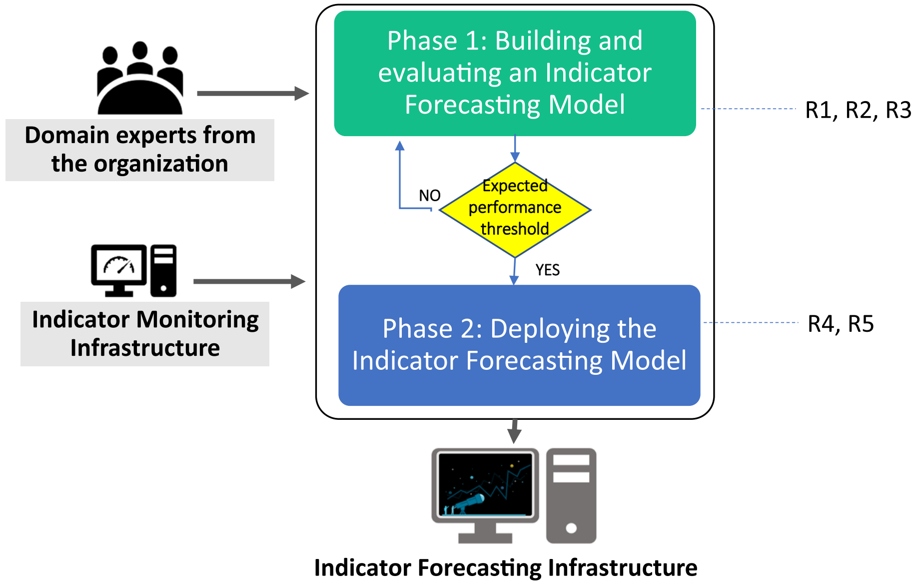

Fig. 4 Overview of the FOSI method

Step 1: *To define the forecasting horizon and the corresponding training and validation sets*. The forecasting horizon defines the length of time into the future for which forecasts are to be computed. Once the forecasting horizon has been defined, the very first action is to split the available software indicator estimation historical data into training and validation sets. The selection of suitable training/validation splits is quite important because as more random variation in the historical data, more data will be needed for obtaining forecasting models that suitably capture such variations (Hyndman and Kostenko 2007). In other words, the more training data is available, the higher the potential performance of the forecasting (Armstrong 2001). Suitable splits depend on the amount of available software indicator estimation historical data and the desired forecasting horizon. Commonly used splits are 70%–30% or 80%–20% for training and validation sets, respectively (Raschka 2018). Hyndman and Athanasopoulos (Hyndman and Athanasopoulos 2018) also recommend that the validation set should be at least as large as the forecasting horizon.

Step 2: *To define the expected performance threshold*. The expected performance threshold refers to the extent to which the intended Indicator Forecasting Model predicts well-founded values. Given the characteristics of the Indicator Forecasting Model to be obtained (see further details in Step 5), for quantifying its performance, we propose the following metrics.

- Accuracy: To determine the percentage of correct predictions made by the Indicator Forecasting Model. It is obtained by calculating the percentage of matches between the forecasted states vs. the estimated states for the validation period.
- F1 score: To determine the Indicator Forecasting Model's balanced ability to both capture positive cases (recall) and be accurate with the cases it does capture (precision). The best value of F1 score is 1 and the worst is 0 (Galdi and Tagliaferri 2018). A high F1 score indicates that the Indicator Forecasting Model is good at predicting both positive and negative cases, which is important for minimizing erroneous predictions.

We suggest these two metrics for evaluating the performance of the Indicator Forecasting Model because accuracy is a reliable metric as long as the dataset has an equal number of samples for each class. However, there would be scenarios where there is no such a balance and leveraging accuracy is not ideal. So, for imbalanced data, F1 score comes to rescue as it considers not just the number of predictions that were incorrect but also the type of errors (false positive and false negative). Hence, the percentage of accuracy and F1 score for the Indicator Forecasting Model should be equal or higher to the expected performance threshold defined.

In addition, as the Indicator Forecasting Model deals with probability distributions, we considered pertinent to complement its evaluation with two additional metrics:

- Jensen-Shannon distance: It measures the distance between two data distributions showing how different the two distributions are from each other (Fuglede and Topsoe 2004). Hence, we applied it to measure the distance between the probability distributions of the forecasted states inferred by the Indicator Forecasting Model and those from the validation set. A value close to zero indicates that the probability distributions are quite similar, thus confirming the accuracy of the forecasted values.
- Base measure contribution error: It aims to quantify the relative importance of the base measure nodes (only those that have a forecasting model) together with their potential prediction error. It is calculated by averaging the difference across the validation set between the Jensen-Shannon distance metric value of the indicator node versus the corresponding Jensen-Shannon distance metric of the base measure node when feeding only its individual forecasted value (i.e., using the prior probabilities of the model for the other base measure nodes).

While the Jensen-Shannon distance metric helps to confirm the confidence on the accuracy of the forecasted values of the nodes, the base measure contribution error metric provides insights on the relative contribution of the node to the potential forecasting errors of the indicator.

Step 3: *Classify base measures.* This step aims to determine how to forecast the base measure values of the Indicator Estimation Model using the training set. Three different types of base measures were devised. For each type, adequate mechanisms to compute its forecasted values for the given forecasting horizon were defined. Base measures can be classified as:

- **Base measures with Known Future Values**: It refers to base measures for which their future values for the forecasting horizon are known or can be estimated by expert knowledge. For instance, thresholds or values related to the allocation of resources. For this type of base measures, the known or estimated values for the forecasting horizon should be stored (for example, in a database) and used as future values.
- **Non-Autocorrelated Base Measures or Base Measure with Missing Values**: It refers to base measures values that do not show any autocorrelation in the training data (i.e., no past values can predict their future behavior) and/or do not have historical data available for computing their forecasts. To support the identification of non-autocorrelated base measure values, we developed the Autocorrelation Test software tool (see Annex 1 for more details) that allows the graphical visualization and/or numerical analysis of a time series dataset to assess its autocorrelation. One way to get the forecasted values of non-autocorrelated base measures and base measures with missing values is to rely on domain's expert judgments. However, if it is not

possible, such values can rely on the prior probabilities of the corresponding base measure nodes from the Indicator Estimation Model. Note that these prior states also come from the domain experts that built the Indicator Estimation Model.

- **Base Measures with Autocorrelated Values**: It refers to base measure values that show autocorrelation in the training data (i.e., the past values of the time series can be used as predictors). Forecasting base measures with autocorrelated values based on the training data led to a time-series forecasting problem. A software companion tool that generates time-series models was developed (see Time Series Gathering tool in Annex 1). We aimed to find a suitable forecasting model for each base measure of this type. The process for finding a forecasting model for each autocorrelated base measure is described in Step 4.

This proposed classification of base measures aims to optimize the effort invested in constructing the Indicator Forecasting Model while addressing real-world dataset challenges. Specifically, it addresses issues such as missing values in base measures, which are critical hurdles when employing forecasting techniques (Schelter et al. 2018), as well as, the need for explainability, which are addressed through the Bayesian nature of the resulting Indicator Forecasting Model.

By classifying base measures based on these three categories, we streamline the process of model building. Base measures with Known Future Values can be directly utilized without the need for forecasting models, as their future values are readily available. Conversely, base measures that lack auto-correlation or contain missing values can leverage the knowledge derived from the Bayesian network of the Indicator Estimation Model. Therefore, only auto-correlated base measures necessitate the development of specific forecasting models. This approach ensures efficient allocation of resources and maximizes the precision of the Indicator Forecasting Model.

Step 4: *To train and select the best forecasting model for each base measure with autocorrelated values.* To automate this step, we decided to implement the hold-out approach suggested by Cerqueira et al. (Cerqueira et al. 2020). So, we developed a support tool called *Hold-out Approach* software kit (see further details in Annex 1). Such kit allows to select the best forecasting model for each autocorrelated base measure. The current version of the kit automatizes 11 forecasting techniques. We made sure that the set of 11 techniques currently implemented covered the three main families of time series forecasting techniques (see Annex 2): 1) *statistical forecasting techniques*: ARIMA, exponential smoothing and decomposition models, 2) *advanced forecasting techniques:* Hybrid forecast models (Shaub and Ellis 2020) and Bagged ETS *(*Bergmeir et al. 2016*)* and 3) *machine learning-based forecasting techniques:* neural networks. The importance of including at least one technique for each family was based on the fact that there is not a single forecasting technique providing the best results for every situation. So, covering the main families of forecasting techniques helps to maximize the chances of finding an adequate forecasting model for any base measure *with autocorrelated values*. Figure 5 provides an overview of the hold-out approach kit implemented to support the FOSI method. It consists of two main processes:

(1) Fitting a candidate forecasting model from each of the forecasting techniques included in the tool using the training set.
(2) To select the best forecasting model. To do so, the set of candidates forecasting models are used to forecast the base measure values for the same period than the validation

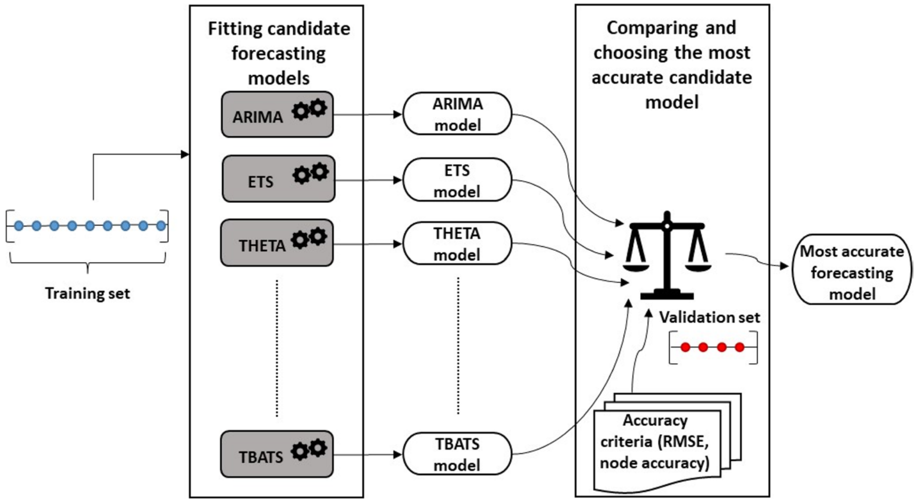

Fig. 5 Overview of the hold-out approach kit software for supporting the FOSI method

set. These forecasted values are then compared with the base measure values from the validation set.

A composite criteria combining two metrics is used to select the best forecasting model: a) accuracy, i.e., the percentage of matches between the forecasted base measure values and the base measure values from the validation set; and b) the Root Mean Squared Error (RMSE) metric, as it is suggested as a suitable metric to compare results from time series forecasting models that are in the same numerical scale (Hyndman and Athanasopoulos 2018; Hyndman and Koehler 2006). The ideal values of the RMSE metric should be zero (meaning that there is no forecasting error). Using this composite criterion, the best forecasting model will be the one yielding the highest accuracy and the lowest RMSE value (prioritizing accuracy).

It is important to remark that the set of forecasting techniques included in the tool, as well as other configuration parameters are flexible and can be extended as required to deal with the forecasting needs of Modeliosoft but also for other similar software companies or purposes.

Step 5: *To articulate the Indicator Forecasting Model and evaluate its performance*. Once the corresponding models/databases that get the forecasted values of each base measure have been built, they should be articulated to ensemble the Indicator Forecasting Model, that is the main output of Phase 1, as shown in Fig. 6.

The Indicator Forecasting Model is composed by the Indicator Estimation Model and the additional models/databases for each corresponding base measure. The Indicator Forecasting Model provides the software indicator forecasting data (i.e., the forecasting of the software indicator, that includes all the states of their composing nodes and their probabilities, as well as the forecasted base measure values).

To evaluate the performance of the resulting Indicator Forecasting Model, the metrics introduced in step 2 are used: accuracy, F1 score, Jensen-Shannon distance and base measure contribution error metrics. The first two metrics aim to be equal or above the expected performance threshold while the last two metrics help to analyze

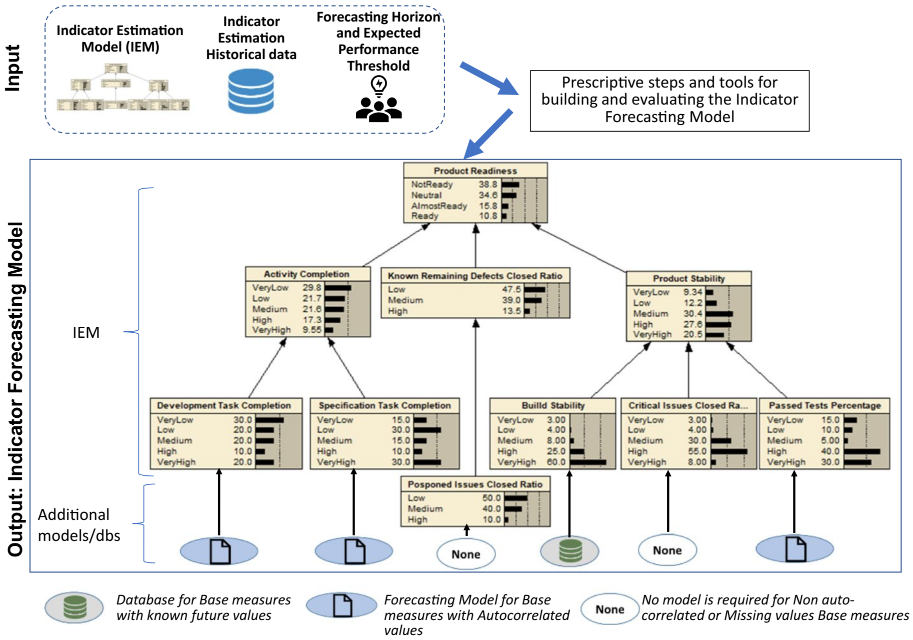

Fig. 6 Summary of Inputs and Outputs of Phase 1

the probability distribution of the Indicator Forecasting Model and identify the base measure nodes that are most prone to error, respectively. To support such analysis, we developed a *Forecasting Report* software tool (see Annex 1) that generates a comprehensive report of the required metrics.

In cases where the expected performance threshold is not achieved, several potential actions should be discussed in order to decide whether:

- To relax the stated performance threshold to proceed with the following phase.
- To promote the repetition of the previous phases with the aim of improving it. For instance, increasing the set of historical data, to adjust the forecasting horizon so that the amount of historical data considered for training the models increases (Hyndman and Athanasopoulos 2018), or to add new forecasting techniques to the *hold-out approach kit*.

Furthermore, the results from the Jensen-Shannon distance and base measure contribution error metrics can provide useful insights to analyze the suitability (mainly in terms of proximity) of the forecasted probability distribution with respect to the validation data or even support the decision of revising the base measure forecasting models (e.g., refit, replace, etc.) that have a high contribution to the error.

5.1.2 Phase 2: Deploying the Indicator Forecasting Model

The decision-support infrastructure of Modeliosoft is based on the Q-Rapids dashboard tool (Q-Rapids Dashboard Tool n.d.) that is a multipurpose web application with a modular, configurable, and extensive architecture. It is composed of the following decoupled components:

- The *Web Application* component receives the user requests from a browser and renders the results through a graphical user interface for monitoring purposes (User Interface component) and orchestrates the different dashboard components (Dashboard Orchestrator component).
- The *Data Collection* component ingests the base measure values from Modeliosoft's corporate repositories through specific data collectors.
- The *Indicator Estimation* component returns the estimations of the software indicator (that is, the discretized states of the software indicator obtained using the unBBayes API to feed the Indicator Estimation Model with the discretized base measures).

The shaded part of the Fig. 7 shows the Modeliosoft's dashboard architecture before adding the forecasting components.

To deploy an Indicator Forecasting Model and enable its integration into the Modeliosoft's dashboard, the ADR Team together with Modeliosoft's software architects extended such existing architecture. Below, we detail the new components, implemented as services, and depicted in the non-shaded part of Fig. 7.

- The *Web Application* component has been extended adding to the *User Interface* component the presentation aspects related to the forecasting requests. Moreover, the *Dashboard Orchestrator* component contains a new *Forecasting Functionality* component that encapsulates the functionalities related to forecasting and relies on the *Indicator Forecasting* component.
- *Indicator Forecasting* component. This new component receives as input a forecasting period and returns the software indicator forecasting data for the given forecasting period. It encapsulates two components:
  - *Indicator Forecasting Wrapper* component. It receives a forecasting period and sends a request to the *Indicator Forecasting Computation* component for calculating the software indicator forecasting data for the given period.
  - *Indicator Forecasting Computation* component. It is in charge of calculating the software indicator forecasting data for the given period. To do so, it queries the corresponding *Additional Databases* that contain the future values from base measures with known future values and from base measures with missing values, when provided by domain experts. The *RServe API* is in charge of obtaining the forecasted values of autocorrelated base measures from the *Files of Base Measure Forecasting Models*. Once the forecasted base measure values have been obtained and discretized to base measure states, they are sent to the *Indicator Estimation Computation* component to feed the corresponding Indicator Estimation Model and return the software indicator forecasting data.

We did not have access to integrate all the components in the production environment of Modeliosoft, instead we executed the resulting architectural components and tools of the FOSI method and tested them in the testing environment of the company. This environment offers a reliable setting similar to the production environment. We did not experience any relevant integration problem, so this architecture was considered suitable as a reference architecture of Modeliosoft's dashboard in production environment.

However, it is crucial to recognize that each software indicator forecasting project is unique and may have specific needs and characteristics. These factors, such as the forecasting demand, data volume, and other contextual considerations, significantly influence

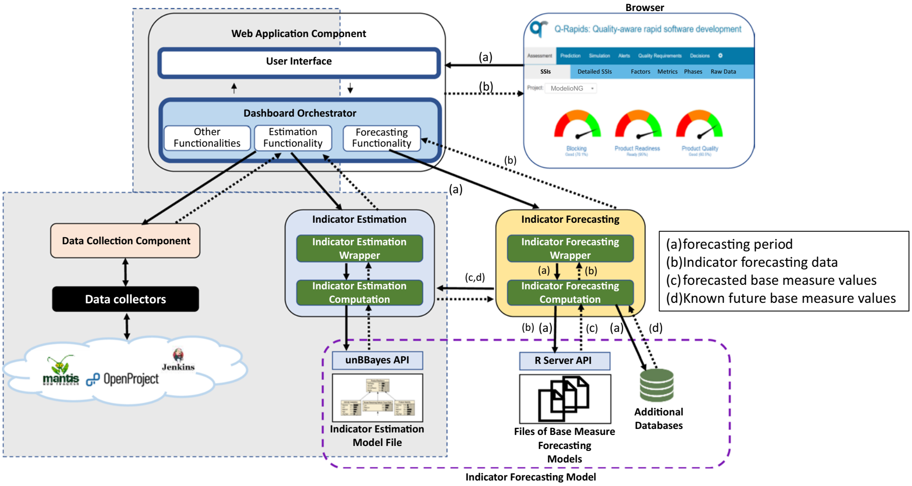

Fig. 7 High-level sketch of a potential integration of the proposed forecasting solution into the Modeliosoft's dashboard

architectural and deployment decisions. Due to this context dependency, this paper does not delve further into these specific decisions. For instance, in cases where the forecasting demand and data involved are substantial, it might be advantageous to employ the *Dashboard Orchestrator* component to manage the storage or materialization of the computed base measure forecasted values into a database. By doing so, subsequent forecasting requests can be processed more efficiently, improving the overall system performance.

The flexible and adaptable nature of the proposed reference architecture allows Modeliosoft to tailor its software indicator forecasting projects according to the specific requirements of each case. As the forecasting needs evolve and new challenges emerge, Modeliosoft can use this reference architecture as a starting point, making necessary adjustments to ensure the most effective and efficient forecasting outcomes for different projects.

5.2 Evaluation of the FOSI Method in a Pilot Project Within Modeliosoft

To evaluate the FOSI method, our focus was primarily on gathering evidence regarding its suitability and technical feasibility.

When referring to the suitability of the FOSI method, we specifically mean its effectiveness in aiding the construction of an accurate and reliable Indicator Forecasting Model. This was addressed through the execution of the activities in Phase I of the method, which aimed to construct and evaluate an Indicator Forecasting Model meeting the expected performance threshold defined by domain experts, as illustrated in Fig. 4.

Regarding technical feasibility, our aim was to assess whether the resulting Indicator Forecasting Model could be successfully operationalized and deployed into Modeliosoft's testing infrastructure. Therefore, the technical feasibility was evaluated based on the success of executing Phase II of the method.

To gather observations on the application of the FOSI method into a pilot project in Modeliosoft, we designed and performed a case study following the guidelines suggested by Runeson et al. (Runeson and Höst 2009). Moreover, we planned an additional feedback evaluation session with Modeliosoft practitioners after the case study execution to gather evidence on the method's usefulness, usability and other relevant aspects that help to its improvement.

It is worth noting that, in this stage, only one representative from Modeliosoft remained actively involved in the ADR Team, specifically the one covering the role of project leader (who is an author of this paper).

The key research question leading the case study is:

"How is the use of the FOSI method in the pilot project?"

The main drivers for the case study design were the phases, steps, tools and procedures suggested by the FOSI method. The case study design was flexible to deal with potential unexpected issues. Details of issues and/or decisions taken during the case study are detailed in the execution section. The Modeliosoft's representative that participated in this stage selected Modelio Wyrm, as a suitable pilot project because:

1) Modeliosoft had an interest on forecasting the *Product Readiness* indicator for this project and were open to share some of their software indicator estimation historical data to run a pilot project.
2) The project had available historical data about the *Product Readiness* indicator (i.e., the estimations of the software indicator, that includes all the states of their composing nodes and their probabilities, as well as the base measure values).
3) The project was being monitored using the existing *Product Readiness* Indicator Estimation Model.
4) He was the project leader of this project, which facilitated the execution of the case study as he acted as our primary contact. Throughout this case study, we will refer to him as the Modelio Wyrm project leader.

Points 2 and 3 are actually a requirement of the FOSI method, as it requires an Indicator Estimation Model and the availability of historical data as inputs.

While such convenient selection may suggest that the chosen pilot project might not fully represent all Modeliosoft projects, it was chosen precisely due to the favorable circumstances discussed earlier. Furthermore, it's worth noting that the primary goal of the company was to obtain preliminary evidence regarding the suitability and feasibility of the FOSI method. Hence, while the chosen pilot project may not fully represent the entirety of Modeliosoft projects, it serves as a valuable exploration into the potential application of the FOSI method within the company.

Modelio Wyrm project leader provided us with all required information about Modelio Wyrm and helped us to shape the software tools (when needed) to fit them to the pilot project needs. Other Modeliosoft's employees related to Modelio Wyrm also participated in the case study but had an indirect contact with the researchers (mainly because of covid-related restrictions).

All issues during the execution of the pilot project were recorded and discussed among all Modeliosoft's participants and subsequently assessed and reflected by the ADR Team.

5.2.1 Modelio Wyrm Project

Modelio Wyrm is the codename of the version 4.0 of Modelio software, an open-source modeling tool providing support for standards like UML, BPMN, ArchiMate, etc. Modelio was first launched in 1991 and it is currently developed and maintained by Modeliosoft.

Modeliosoft Wyrm development process is defined as scrum-like, "close to agile" with no pre-defined sprint cycles, with a strong focus on their current issues, the final product, and close cooperation with the customers. The tools used by the development team were: *OpenProject*[7] for project management (backlog management, issues, and specification tracking), *Mantis*[8] for bug tracking and *Jenkins*[9] for building and testing.

Modelio Wyrm is a strategic project for the company and its *Product Readiness* indicator was being monitored and visualized through the corporate dashboard. It implies that there exists a *Product Readiness* Indicator Estimation Model, the corresponding data collectors and historical data available.

5.2.2 Data Collection and Data Analysis

For security and confidentiality reasons, we did not have direct access to the company repositories, but the Modelio Wyrm project leader provided us with a snapshot of 47 days of historical data about the *Product Readiness* indicator, corresponding to a specific release under development of Modelio Wyrm. Such data was obtained automatically from data collectors. These data collectors extracted and stored information from Modeliosoft's corporate repositories (mainly development process/product-related repositories) on a daily basis. The data was stored in an Elasticsearch[10] node, a document-based database.

Data analysis was performed according to the phases and steps of the FOSI method. We also used individual diaries to record notes on any type of issues and aspects that we considered relevant (e.g., problems, schedule, effort, decisions, attitudes). Improvements and adaptations done to the software tools together with their rationale were also recorded. We provide as much detail as possible given non-disclosure agreements with Modeliosoft.

In addition, we designed a survey based on a questionnaire as a data collection instrument for gathering feedback from Modeliosoft's practitioners that participated in the pilot project about the usefulness and usability of the FOSI method.

5.2.3 Case Study Execution

The project leader of Modelio Wyrm involved five people from his Modelio Wyrm's team in the execution of the pilot project. He wanted them to have first-hand contact with the FOSI method in order to know their impressions about it after the execution of the pilot project. So, the project leader provided a general explanation (avoiding technical details) of the phases and steps of the FOSI method before the execution of the case study and also informed the goals of the pilot project.

Phase 1: Building and Evaluating the Product Readiness Forecasting Model.

Step 1: To define the forecasting horizon and the corresponding training and validation sets. Based on the available historical data and the forecasting needs of the project, the forecasting horizon was set to 14 days. The Modelio Wyrm project leader stated that this forecasting horizon was meaningful to provide enough room to prevent risk situations for the specific release under development of Modelio Wyrm. The software indicator estimation historical data was split into 33 days for the training set (70% of the total) and the remaining 14 days (corresponding to the forecast horizon) were used for the validation set (30% of the total).

[7] https://www.openproject.org

[8] https://www.mantisbt.org

[9] https://jenkins.io

[10] https://www.elastic.co

Step 2: *To define the expected performance threshold*. Modelio Wyrm's project leader provided confirmation that, at the time, the project was not facing a critical situation. Given this assessment, he made the decision to establish the forecasting performance threshold at 70%. This threshold represents the level of performance deemed acceptable or satisfactory for the project's forecasting efforts. By setting this threshold, the project leader aimed to ensure that the forecasting model would provide reliable predictions while allowing for a margin of error or variability in the forecasts. This decision reflects a balanced approach to forecasting performance, considering the project's current status and objectives.

Step 3: *Classify base measures.* The autocorrelated base measure values of the training set were collected from the Elasticsearch database and modeled as standard time series in R, using the software tool *Time Series Gathering*. The *Autocorrelation Test* tool helped to confirm if it was autocorrelation or not. Table 5 summarizes the classification of each base measure of the *Product Readiness* Estimation Model and its rationale.

Step 4: *To train and select the best forecasting model for each base measure with autocorrelated values*. For each base measure with autocorrelated values, we used the *Hold-Out Approach kit* for fitting and selecting its best forecasting model. Table 6 summarizes the resulting values of the metrics used by the *Hold-Out Approach kit* (i.e., accuracy and RMSE) for each one of the models used. In green it is shown the selected model for each base measure (i.e., the one that had the best results from accuracy and RMSE).

The obtained results were quite positive. We did not have any problem related to finding suitable forecasting models. Therefore, we could proceed with Step 5.

Step 5: *To articulate the Indicator Forecasting Model and evaluate its performance.* The forecasting models/databases for each corresponding base measure were integrated with the *Product Readiness* Estimation Model to obtain the resulting *Product Readiness* Forecasting Model, as shown in Fig. 8.

To evaluate the performance of the resulting *Product Readiness* Forecasting Model, we used the validation set and calculated the suggested metrics as shown in Table 7. To obtain such a report, we used the *Forecasting Report* tool.

We can observe that all individual nodes fulfill the expected performance threshold (70%) for accuracy and F1 score. More precisely, the accuracy of the *Product Readiness* indicator node was 71.43%, and F1 score was 83%.

In addition, the values of the Jensen-Shannon distance metric of all nodes are quite low. This confirms that the distances between the forecasted states inferred by the Indicator Forecasting Model and those from the validation set were quite low, indicating the similarity of the probability distributions. On the other hand, the results of the computation of the *base measure contribution error* for each base measure node shows that the base measure *Dev. Task Completion* got the highest value for this metric (0.35). It means that this is the base measure node with highest relative importance in the whole model or it is the most prone to error.

Overall, the findings presented in Table 7 demonstrated the effectiveness of the *Product Readiness Forecasting Model* for Modelio Wyrm, as derived from the application of the FOSI method. The *Product Readiness Forecasting Model* successfully forecasted the values of its nodes within the predetermined threshold set by the Modelio Wyrm project leader. Consequently, the project leader endorsed the suitability of the resulting forecasting model for its deployment in Phase 2.

Phase 2: Deploying the Product Readiness Forecasting Model After completing the previous phase and achieving successful results regarding the suitability of the *Product Readiness Forecasting Model*, the model was deemed ready for deployment. Following the reference architecture of Modeliosoft's dashboard, as defined in the FOSI method (see Fig. 7), we successfully instantiated the corresponding *Files of Base Measure Forecasting Models* and *Additional*

Table 5 Classification of Base Measures, Rationale and Storage

Categorization	Base Measure	Rationale	Obtention and/or Storage of Forecasted Values
Base Measure with Autocorr. Values	-Dev. Task Compl. -Spec. Task Compl. -Passed Tests Perc.	Their autocorrelation was confirmed by the information provided by the *Time Series Gathering* and the *Autocorrelation Test* tools.	The forecasted values are obtained through their corresponding forecasting model (to be built in step 4 below).
Base Measures with Missing Values	-Post. Issues (Closed) Ratio -Critical Issues (Closed) Ratio	In both cases there was no data for the period stated in the training data. They were not collected during the training period for an unexpected problem in the corresponding data collectors.	The forecasted values are obtained from the prior *Product Readiness* Indicator Estimation Model probabilities.
Base Measure with Known Future Values	-Build Stability	Modelio Wyrm project leader confirmed (after discussing with the development team) that the expectation for this base measure would be to keep its constant value for the forecasting horizon (i.e., any change was expected in the percentage of successful builds with respect to the total of builds triggered in each seven-day period).	The constant value for the forecasting horizon was stored in a physical storage (i.e., a csv file).

Table 6 Results of the hold-out approach for each autocorrelated base measures

Type		Statistical							Advanced			ML
Base Measure	Criterion	ARIMA	ARIMA FS	THETA	ETS	ETS DM	STL	TBATS	BAG. ETS	HYBRID	PROPHET	NN
Development Task Completion	Accuracy (%)	78.6	14.2	64.2	50	78.6	78.6	14.2	50	7.1	0	78.6
	RMSE	0.090	0.154	0.067	0.108	0.073	0.063	0.430	0.083	0.445	0.206	0.076
Specification Task Completion	Accuracy (%)	78.6	78.6	78.6	78.6	78.6	78.6	78.6	78.6	78.6	28.5	78.6
	RMSE	0.107	0.092	0.062	0.089	0.089	0.091	0.113	0.073	0.453	0.154	0.089
Passed Tests Percentage	Accuracy (%)	78.6	14.2	50	78.6	78.6	0	78.6	0	71.4	78.6	78.6
	RMSE	0.146	0.258	0.054	0.06	0.063	0.27	0.065	0.259	0.056	0.08	0.179

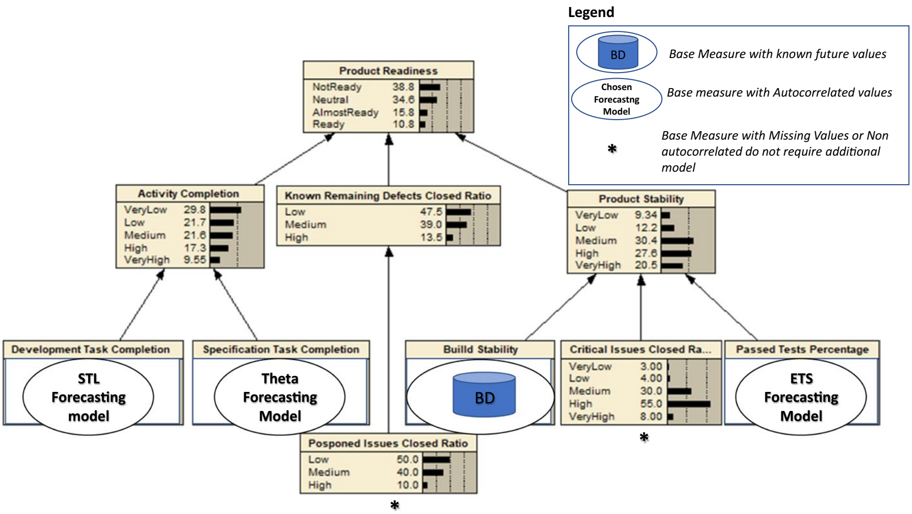

Fig. 8 Resulting Product Readiness Forecasting Model

Databases in a test environment, and connected them with their respective components. As a result, the initial release of the forecasting functionality was integrated into Modeliosoft's dashboard. While this initial functionality provided rudimentary forecasting visualizations, all participants in the Modeliosoft case study were able to navigate through the forecasted values associated with the *Product Readiness* indicator. Given their familiarity with Modeliosoft's dashboard and previous estimation functionalities (Manzano et al. 2021), they encountered no difficulties in exploring the forecasting capabilities. The Modeliosoft's representative expressed his belief that offering forecasting capabilities would enable them to proactively address potential issues or capitalize on opportunities. Informal feedback from the case study participants included remarks such as: "These forecasts are highly relevant in supporting our decision-making processes" and "As a project leader, I am keen on monitoring how the baseline measures evolve over time. Thus, visualizing these forecasted values enables me to make corrective decisions based on them".

5.2.4 Feedback Evaluation

The ADR Team designed an interview-guide instrument aimed to gather the perceptions about the usefulness and usability of Modeliosoft's practitioners that participated in the pilot project in order to improve the FOSI method. We used the guidelines stated by Oates (2006). The guide included mainly 3 open questions related to a) positive/negative perceptions about the forecasting solution and its execution, b) suggestions and improvements and c) additional comments.

Table 7 Performance metrics computation for Product Readiness and its nodes

Result/ Node Name	Dev. Task Compl.	Spec. Task Compl.	Passed Tests Percent.	Activity Compl.	Known Rem. Defects (Closed) Ratio	Product Stability	Product Readiness
Node Type	Base measure			Derived measure			Indicator
Matches (forecasted states vs validation set)	11/14	11/14	11/14	11/14	10/14	12/14	10/14
Node accuracy (%)	78.57	78.57	78.57	78.57	71.43	85.71	71.43
F1 score (%)	88	88	88	88	83	100	83
Jensen-Shannon distance	0.18	0.18	0.18	0.07	0.11	0.04	0.05
Base Measure Contribution Error	0.35	0.17	0.09	–	–	–	–

We planned to perform semi-structured interviews with the five Modeliosoft's practitioners that participated directly on the activities related to the pilot project, but given some organizational restrictions due to Covid-19, we replaced our interview plan. We finally adapted the interview-guide instrument to be used as a guide for a focus group session led by the Modelio Wyrm project leader. He has previous experience on this type of sessions, so he did not require additional training. We recommended some actions to mitigate the participants' evaluation apprehension. For instance, remarking at the beginning of the session, that the goal of the case study and the focus group was not to evaluate the employees but to assess the feasibility of the method's application. Therefore, their honest opinions (without enhancing/hiding issues) were expected.

The five employees that participated closely on the activities of the pilot project attended to the focus group. The Modelio Wyrm project leader provided us with his notes from the focus group. We, as researchers, are aware of the threats associated to the fact that we did not participate directly on the focus group, as it increases the threats of using subjective and not contextualized opinions. However, the feedback received included both positive aspects and issues to be improved that widely fit with our own notes and observations taken during the execution of the case study. The main aspects raised by the participants were:

- The FOSI method was prescriptive enough. The steps described to put forward the forecasting proposal were mainly perceived as straightforward and feasible by Modeliosoft employees that participated closely on the activities of the pilot project. One of them commented that the steps described in the solution could be repeatable in other projects as long as they have all the supporting software artifacts and the potential availability of an expert to guide/support some of their decisions in case they need it.
- Availability of software support tools for the steps of the FOSI method was the most valued aspect. Employees agreed that without the supporting tools, they would not be able to build the Indicator Forecasting Model. Employees mentioned that the availability of the software tools saves extensive manual effort and significantly hides the complexity of building a forecasting model.
- Reuse of existing infrastructure was positive. The use of the existing Modeliosoft infrastructure was perceived as a very positive aspect as they were already familiar with it. On the one hand, the fact that the integration of the resulting Indicator Forecasting Model into their existing testing infrastructure was prescribed as a phase of the FOSI method was positively valued by the practitioners. On the other hand, the fact that the Modeliosoft dashboard was used to visualize the forecasting results was considered a factor that would promote the rapid adoption of the forecasting functionality by the corresponding practitioners (without requiring additional training as they are already familiar with the Modeliosoft dashboard usage).
- Need of a basic training program about forecasting principles to better understand the rationale behind the FOSI method steps. Some employees mentioned that they follow the prescribed activities but they do not really understand their rationale. This coincide with our notes taken during the execution of the pilot project, as we experienced several questions and doubts mainly related to the steps of Phase I (i.e., the construction of the Indicator Forecasting Model). It uncovered the need of promoting a basic training program in the company, so the corresponding employees are aware of basic concepts that help them to better understand the FOSI method's rationale.

Among the potential improvements discussed within the ADR Team, two noteworthy ones are being implemented in Modeliosoft:

- *Improved documentation with real examples:* Recognizing the significance of clear and practical documentation, the team worked on enhancing the existing documentation. The improved documentation now includes real examples derived from Modeliosoft's specific use cases and applications of the Indicator Forecasting Model. These real examples serve as concrete illustrations to guide employees and facilitate their understanding of the FOSI method's application in actual scenarios within the company.
- *Basic forecasting training program:* The team agreed to develop a basic forecasting training program tailored to the needs of the employees who would be applying the method. This training program aims to equip them with the necessary knowledge and skills to effectively use the FOSI method. The training will cover essential concepts, techniques, and best practices in forecasting, ensuring that employees are well-prepared to utilize the method proficiently.

6 ADR: Formalization of Learning

Aligned with the ADR principle of *Generalized Outcomes* (P7), the FOSI method was conceptualized as a forecasting solution for Modeliosoft, while offering insights that may be applicable to other organizations facing similar challenges. To do so, we followed the generalization efforts suggested by ADR:

1) Generalization of the problem instance. We approached the forecasting problem at Modeliosoft as an instance of a broader class of issues. In evaluating Modeliosoft's forecasting needs, we also considered similar forecasting challenges that other organizations might encounter (see Section 4.2).
2) Generalization of the solution instance. During the design of the FOSI method the ADR team took deliberated decisions to maximize its potential applicability. Some of the most relevant ones are:

    - *Decoupled phases:* To enhance adaptability and flexibility in addressing diverse organizational needs, the FOSI method was structured into two distinct phases. Each phase delineates clear inputs, outputs, and procedural steps, facilitating their independent application. Specifically, Phase I addresses reusability by outlining the construction of an Indicator Forecasting Model based on state-of-the-art forecasting techniques, and offering suggested steps and tools. We believe that this can foster the industrial uptake of existing forecasting techniques by inspiring other software-development organizations on how to deal with similar problems, and even reuse the procedures and software support tools resulted from this work (as they were released as open source). Conversely, Phase 2 relies heavily on Modeliosoft's infrastructure, potentially limiting its reusability. To mitigate this limitation, we provided a comprehensive explanation of the architectural components, enabling readers to understand their applicability beyond the confines of Modeliosoft. Additionally, the utilization of Modeliosoft's dashboard architecture, which is based on the freely available Q-Rapids dashboard (Q-Rapids Dashboard Tool n.d.), further enhances its potential for widespread adoption.
    - *Generic software support tools:* The software tools developed to support the FOSI method were designed to be generic and adaptable, ensuring their potential for

reuse across different organizational contexts. Furthermore, these tools are released as open-source software, fostering their accessibility and adoption.

3) Generation of design principles from the experience of developing and applying the FOSI method. Design principles reflect knowledge of both IT and human behavior (Gregor et al. 2020) and should provide cues about the goal that need to be achieved and how to achieve it, so the target audience can better understand how to apply the FOSI method. Table 8 outlines the main design principles envisaged by the ADR Team.

All in all, we believe that our efforts can foster the industrial uptake of existing forecasting techniques by inspiring other software-development organizations on how to deal with similar forecasting problems, and even reuse the procedures and software support tools resulted from this work (as they were released as open source).

7 Limitations and Threats to Validity

Below we detail the actions performed to strength the rigor, trustworthiness, credibility, and transferability of our results (Ralph et al. 2020).

- *Rigor:* We applied the ADR method in a rigorous way, keeping a transparent account of the research process, including the documentation of research activities, and the rationale of the decisions made by the ADR Team. This transparency enhances the rigor of the research and allows for further scrutiny by other researchers.
- *Transferability:* We are aware that the proposed method and its validation are specific to Modeliosoft context, however, the design of the method was influenced by the aim of formulating generic-and-abstract recommendations that help other organizations facing similar problems. In addition, we would like to emphasize that we tried to describe the whole design of the industry-academia collaboration and provide as many details as possible (while reconciling confidentiality issues) of the problems, requirements and infrastructure of Modeliosoft. Our main aim was that other companies can identify their potential similarities and get inspired by the solution proposed in Modeliosoft. In addition, the software support tools developed have been released as open source, so others can adapt and use them for similar purposes.
- *Participant Bias.* Since the beginning of the study, the ADR Team had a clear understanding of the objective of the collaboration: improving Modeliosoft forecasting capabilities by working together to generate and apply useful knowledge for both the company and researchers, which leads to a 'win-win' scenario. To avoid researchers bias, we conducted the study under the premises of *impartiality,* considering all data gathered and expressed by all participants, and promoting *continuous reflexivity* (individually and as a team) to make sure that all data is assessed with a clear and unbiased mind. So, we used to share and continuously discuss our impressions and notes to keep pre-existing assumptions at bay. Our actions taken to minimize the potential bias from the case study's participants are reported in Section 5.2.4 (e.g. evaluation apprehension, non-direct feedback from the focus groups).
- *Smooth collaboration.* The fact that we had previous involvement with Modeliosoft in previous industry-academia collaborations meant an opportunity (instead of a problem), as

Table 8 Summary of Design Principles resulted from this research

Design Principle	Goal to be achieved	How to achieve the goal
1.- To inform decisions based on automatic forecasting of software indicators requires the availability of domain expert knowledge and corporate repositories information.	-Building a Forecasting Indicator Model that deals with partial missing values of corporate repositories and provide explainability. (R1 and R2)	-Departing from an Indicator Estimation Model, and historical information, Phase I of the FOSI method provide guidance to build an *Indicator Forecasting Model*. Detailed guidance on how to build an *Indicator Estimation Model* for those organizations that do not have it already, is provided in (Manzano et al. 2021).
2.- Most of the technical knowledge required to build and operationalize a Forecasting Indicator Model can be found in software support tools.	-To minimize the specialized knowledge required to build an Indicator Forecasting Model. (R3)	-The procedures detailed in Phase I of the FOSI method are supported with open source software tools that allows the construction and operationalization of an *Indicator Forecasting Model*.
3.-Additional forecasting techniques can be added to succeed on the construction of suitable Indicator Forecasting Models.	-To expand the scope of forecasting techniques implemented within the FOSI method, thereby providing a more comprehensive forecasting solution. (R3)	-To add new forecasting technique(s) as a software component of the Hold-Out Approach Kit (see Annex 1).
4.- The deployment of the Indicator Forecasting Model requires a robust infrastructure that allows data processing and visualization.	-To integrate the Indicator Forecasting Model into the decision-making support infras-tructure of the organization. (R4 and R5)	- Phase II of the FOSI method delves into the architectural components specific to Modeliosoft. Such architecture is based on the Q-Rapids dashboard, a freely available tool (Q-Rapids Dashboard Tool n.d.). Therefore, the insights gleaned from this phase can serve as a source of inspiration, or alternatively, organizations may opt to utilize the Q-Rapids dashboard.

we were familiar with its context and had built a smooth collaboration environment that undoubtly influenced this project positively. In addition, Modeliosoft's representatives were convinced on their goal of promoting forecasting capabilities and fully supported all related activities of this collaboration. As reported before, one of the representatives could not manage to participate in the last stages of the ADR collaboration (Stages 3–4, due to lack of time to process all his duties). However, this did not affect the reciprocal knowledge sharing of the ADR Team as his knowledge was mostly vital in the stages that he/she participated. Furthermore, the other representative was available and willing to provide us any required clarification or feedback for the smooth execution of the remaining stages.

- *Online interactions:* Given Covid-19 restrictions, most of our interactions were held online instead of face-to-face in the company premises. We are aware that this could have a negative influence on the communication abilities and therefore on our results. However, the flexibility of our collaboration plan led us to adapt to the corresponding restrictions and get the best results as possible. In addition, the fact that we had previous face-to-face meetings with whole team in previous projects (so we knew each other and built in a previous confidence on our commitment, capabilities and trustwothiness) helped to minimize the effects of mostly online interactions. In addition, the ADR stages mostly focused on the technical feasibility of the method, and not on the organizational perspective (that would require more social interaction and would be further affected by the online issues).

8 Lessons Learned

This section provides insights into the lessons learned from this industry-academia collaboration.

Our previous ventures in industry-academia collaboration had primarily centered on either Design Science (DS) or Action Research (AR). However, in this work, we uncovered the potential of Action Design Research (ADR) as a methodologically robust framework that allows the integration of elements from AR, DS, and technology transfer models for covering the objectives of our industry-academia collaboration (as detailed in Section 3). In particular, the ADR's ability to facilitate artifact design, co-creation, and evaluation within the same industrial setting (Modeliosoft) was invaluable for rigorously obtaining swiftly actionable results from Modeliosoft. Although preliminary, the obtained results addressed Modeliosoft's need to rapidly evaluate the viability of the indicators forecasting idea while provided us with initial design knowledge to be further explored in future industry-academia endeavors. Some relevant aspects on the use of ADR are highlighted below.

- Promotion of co-creation: The collaborative nature of ADR fostered co-creation between the researchers from UPC and representatives from Modeliosoft. Together, we worked to envision the FOSI method, where both parties contribute to and benefit from the knowledge created. This collaborative effort positively influenced team motivation and involvement. We observed a more natural and genuine involvement of Modeliosoft representatives than when using other methods that do not emphasize co-creation.
- Short collaboration cycles. In essence, this cycle was dedicated to envisioning a forecasting solution for Modeliosoft, known as the FOSI method, and evaluating its adherence to the specified requirements outlined by Modeliosoft. The cycle specifically focuses on the method's suitability and technical feasibility, rather than delving into potential improvements to Modeliosoft decision-making processes, which is planned

to be addressed in subsequent collaboration cycles. This cycle prioritized assessing the effectiveness and practicality of the FOSI method for Modeliosoft immediate needs. By focusing on the core objectives of each cycle, Modeliosoft and the researchers can make prompt and realistic decisions regarding the continuation of the collaboration. In our experience, short collaboration cycles have been instrumental in driving the success of our collaboration with Modeliosoft, facilitating rapid progress and efficient decision-making. This finding aligns with the conclusions drawn by Mikkonen et al. regarding the suitability of planning short cycles for technology transfer projects between industry and academia (Mikkonen et al. 2018). Furthermore, the inherent ADR flexibility allowed us to adapt to such short collaboration cycle under the resources and restrictions of Modeliosoft. This was especially important as we faced unexpected Covid related issues and had to be flexible on the planned feedback evaluation (as detailed in Section 5.2.4.)

- Evaluation of the ensembled artifact (i.e., the FOSI method) within the organization: This was a positive ADR aspect for both researchers and Modeliosoft as it in some way expedited the evaluation process and therefore the obtention of prompt evidence.
- Reconciling confidentiality issues of the company with the goal of supporting similar field problems posed a significant challenge. Finding a balance between safeguarding proprietary information and leveraging shared knowledge required extensive discussions and reviews by the ADR Team. Our primary concern was to ensure that sensitive details about Modeliosoft operations or proprietary technologies were not inadvertently disclosed. The use of ADR was convenient for this matter as it suggests to describe the ensemble artifact in an abstract manner. This allowed us to discuss and develop solutions without divulging sensitive information. While this level of abstraction was natural for Phase 1 of the FOSI method, it posed a challenge for Phase 2, which required describing architectural details. However, the fact that the architecture was based on the Q-Rapids dashboard (Q-Rapids Dashboard Tool n.d.) which is freely open, proved to be advantageous. Leveraging the Q-Rapids dashboard architecture enabled us to discuss architectural components without revealing proprietary information.
- Shared ownership. Our co-creation process led us to benefit from the knowledge created (as long as it does not involve sensitive information from the company). This approach enabled us to improve and use the FOSI method artifacts independently outside the industry-academia collaboration. The fact that the FOSI method (especially Phase 1) was designed to be abstract and reusable favor this. Currently, researchers are focusing on enhancing the FOSI method by incorporating new forecasting techniques. These enhancements aim to broaden the method's versatility and applicability, enabling it to address a wider range of forecasting problems. Importantly, these improvements will be made freely available so that not only Modeliosoft but also others can benefit from them. This commitment to shared knowledge underscores the collaborative spirit of the industry-academia partnership and promotes the advancement of research and practice in the field.

Despite the mentioned aspects, the most burdensome aspect of applying ADR was:

- Lack of guidance on ADR application and reporting. While the flexibility of ADR is appreciated, its absence of clear guidance hindered the elucidation of procedural steps in certain stages. Particularly, we found a dearth of direction or references for formulating design principles during the Formalization of Learning stage, as well as for reporting the work. This absence made it challenging to navigate through these stages effectively. We hope that more efforts will be done to support ADR's adoption in SE as it resulted a very good option to address industry-academia collaborations.

It is important to emphasize that our initial collaboration with Modeliosoft through a European H2020 financed project played a pivotal role in shaping our relationship. Our smooth collaboration experience, coupled with the ability to detail the architecture of the Modeliosoft dashboard (built upon the freely available Q-Rapids dashboard (Q-Rapids Dashboard Tool n.d.)), underscores the significant impact of such financed projects in fostering subsequent collaborations and co-creation efforts. We attribute much of the success of this industry-academia collaboration to the foundation laid during the Q-Rapids project (H2020 EU project - Quality Aware Rapid Software Development project n.d.).

Finally, after our experience with ADR, we suggest to complement Wohlin and Runeson's list of methodologies for dealing with industry-academia collaborations in software engineering (Wohlin and Runeson 2021) with ADR. Wohlin and Runeson delineate the primary objectives of these methodologies as follows: a) DS focuses on acquiring design knowledge through the design of artifacts; b) AR centers on change in socio-technical systems; c) technology transfer models prioritize the transfer of research to industry. Our suggested primary objective for ADR is on acquiring design knowledge through the co-creation and evaluation of an artifact that brings a change in a socio-technical system, where such co-creation and evaluation occurs in the same organization. This unique characteristic of ADR, promoting both artifact creation and evaluation within the same organization holds significant potential within software engineering contexts, as it might help to deal with some well-known challenges associated with securing industrial participation (Basili et al. 2018; Song and Runeson 2023; Wohlin and Runeson 2021).

9 Conclusions and Future Work

This paper outlines the outcomes of an industry-academia collaboration aimed at facilitating software indicators forecasting within Modeliosoft, a private software-development organization. Following Action Design Research (ADR) principles, the collaboration led to the development and evaluation of a method named FOSI (**F**orecasting **O**f **S**oftware **I**ndicators) within Modeliosoft. Additionally, the paper introduces four design principles intended to assist other organizations grappling with similar forecasting challenges as those faced by Modeliosoft. By elucidating the intricacies of this industry-academia collaboration and its results, the paper aims to promote the realism and applicability of software engineering research. Moreover, it endeavors to underscore the mutually beneficial nature of industry-academia collaborations.

Our ongoing efforts with Modeliosoft are geared towards fostering the next short-term collaboration cycle, which could potentially focus on bridging the Indicator Forecasting Models produced by the FOSI method with actual Modeliosoft decision-making processes, while also assessing their organizational impact.

Annex 1

Table 9 Tools for supporting and automating processes of the FOSI method

Step	Software tool description	Input	Output
3	**Time Series Gathering Tool** (Manzano 2021a): Generates a time series model in R (R. Development Core Team, R 2008) from a dataset extracted from a database[a].	-Database parameters -Dataset -Frequency -Training period	-Dataset modeled as a time series
3	**Autocorrelation Test Tool** (Manzano 2021): Assess autocorrelation graphically and numerically based on (Hyndman and Khandakar 2008) and (Hyndman and Athanasopoulos 2018).	-Time series dataset	-Graphical/numerical confirmation of autocorrelation.
4	**Hold-Out Approach Kit:** It wraps and uses Model Fitting, Forecasting Execution, and Model Comparison tools for automatically applying the hold-out approach.		
	Model Fitting Tool (Manzano 2021) (available for the forecasting techniques shown in Annex 2): Fits a forecasting technique with a time series training dataset to obtain a fitted model with the selected technique.	-Training dataset -Forecasting technique	-Fitted model
	Forecasting Execution Tool (Manzano 2021b): Executes a forecasting model with a given forecasting horizon to obtain the corresponding forecasted values.	-A forecasting model -Forecasting horizon	-Forecasted values
	Model Comparison Tool (Manzano 2022a): Compares forecasting models and selects the best one based on accuracy and Root Mean Squared Error metrics.	-Validation set -Forecasting models	-Most accurate forecasting model
5	**Accuracy Computation Tool** (Manzano 2022b): Computes the accuracy of the Indicator Forecasting Model.	-Validation set -Indicator Forecasting Model	-Accuracy of the Indicator Forecasting Model
5	**Distance Computation Tool** (Manzano 2021c): Computes the Jensen-Shannon distance (Haj-Bolouri et al. 2018; Fuglede and Topsoe 2004) between two probability distributions datasets.	-Validation set -Forecasted states of the indicator	-Average distance between the forecasts and the validation set
5	**Forecasting Report Tool (Manzano** 2022c): Generates a CSV report from the results obtained from the tools above.	-Results from the tools used in the previous steps	-Comprehensive report

[a]It uses an Elasticsearch database but can be easily changed to other technology

Annex 2

Table 10 Forecasting techniques used to devise base measure forecasting models

Type	Acronym	Full Name/Reference
Statistical	ARIMA	Autoregressive Integrated Moving Average (Newbold 1983)
	ARIMA FS (forcing seasonal models)	Autoregressive Integrated Moving Average forcing seasonal models (Newbold 1983)
	THETA	Theta model: a decomposition approach to forecasting (Assimakopoulos and Nikolopoulos 2000)
	ETS	Exponential Smoothing State Space Model (Hyndman et al. 2008)
	ETS DM (forcing damped models)	Exponential Smoothing State Space Model forcing damped models (Hyndman et al. 2008)
	STL	Seasonal Decomposition of Time Series by Loess (Cleveland et al. 1990)
	TBATS	TBATS model (Exponential smoothing state space model with Box-Cox transformation, ARMA errors, Trend and Seasonal components) (De Livera et al. 2011)
Advanced	BAGGED ETS	Bagged ETS (Bergmeir et al. 2016)
	HYBRID	Hybrid Forecast Models (Shaub and Ellis 2020)
	PROPHET	Prophet (Taylor and Letham 2018)
Machine Learning	NN	Feed Forward Neural Networks (Hyndman and Khandakar 2008; Tadeusiewicz 1995)

Acknowledgments This work was partially supported by the "Spanish Ministerio de Ciencia e Innovación" under project / funding scheme PID2020-117191RB-I00 / AEI/https://doi.org/10.13039/501100011033.

Author contribution

• Claudia Ayala: Conceptualization, Methodology, Formal Analysis and Investigation, Project Administration, Supervision, Writing - Original Draft, preparation, Writing - Review and Editing.

• Cristina Gómez: Conceptualization, Methodology, Formal Analysis and Investigation, Project Administration, Supervision, Writing - Original Draft, preparation, Writing - Review and Editing.

• Martí Manzano: Conceptualization, Formal Analysis and Investigation, Software, Writing - Original Draft.

• Antonin Abherve: Resources, Investigation, Software.

• Xavier Franch: Funding acquisition, Writing - Review & Editing.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work has been partially funded by the Spanish Ministerio de Ciencia e Innovación under project/funding scheme PID2020-117191RB-I00/AEI/10.13039/ 50 501100011033.

Data Availability The datasets used in this study are not available due to explicit request from the participating company. But all supporting tools are available in their corresponding repositories stated in the list of references of this paper.

Declarations

Conflicts of Interest/competing Interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

Amin A, Grunske L, Colman A (2013) An approach to software reliability prediction based on time series modeling. J Syst Softw 86:1923–1932. https://doi.org/10.1016/j.jss.2013.03.045

Ardimento P, Aversano L, Bernardi ML, Cimitile M, Iammarino M (2022) Using deep temporal convolutional networks to just-in-time forecast technical debt principal. J Syst Softw 194:111481. https://doi.org/10.1016/J.JSS.2022.111481

Armstrong JS (2001) Principles of forecasting. Springer US, Boston, MA. https://doi.org/10.1007/978-0-306-47630-3

Assimakopoulos V, Nikolopoulos K (2000) The theta model: a decomposition approach to forecasting. Int J Forecast 16:521–530. https://doi.org/10.1016/S0169-2070(00)00066-2

Avison DE, Lau F, Myers MD, Nielsen PA (1999) Action research. Commun ACM 42:94–97. https://doi.org/10.1145/291469.291479

Baltes S, Ralph P (2022) Sampling in software engineering research: a critical review and guidelines. Empir Softw Eng 27:94. https://doi.org/10.1007/s10664-021-10072-8

Basili V, Briand L, Bianculli D, Nejati S, Pastore F, Sabetzadeh M (2018) Software engineering research and industry: a symbiotic relationship to Foster impact. IEEE Softw 35:44–49. https://doi.org/10.1109/MS.2018.290110216

Bergmeir C, Hyndman RJ, Benítez JM (2016) Bagging exponential smoothing methods using STL decomposition and box–cox transformation. Int J Forecast 32:303–312. https://doi.org/10.1016/J.IJFORECAST.2015.07.002

Bosch J (2016) Speed, data, and ecosystems: the future of software engineering. IEEE Softw 33:82–88. https://doi.org/10.1109/MS.2016.14

Cerqueira V, Torgo L, Mozetič I (2020) Evaluating time series forecasting models: an empirical study on performance estimation methods. Mach Learn 109:1997–2028. https://doi.org/10.1007/s10994-020-05910-7

Chambers JC, Mullick SK, Smith DD (1971) How to choose the right forecasting technique. https://hbr.org/1971/07/how-to-choose-the-right-forecasting-technique. Accessed 29 Apr 2021

Chazette L, Schneider K (2020) Explainability as a non-functional requirement: challenges and recommendations. Requirements Eng 25:493–514

Choetkiertikul M, Dam HK, Tran T, Ghose A, Grundy J (2018) Predicting delivery capability in iterative software development. IEEE Trans Softw Eng 44:551–573. https://doi.org/10.1109/TSE.2017.2693989

Chug A, Malhotra R (2016) Benchmarking framework for maintainability prediction of open source software using object oriented metrics. Int J Innov Comp Inf Control 12(2):615–634

Cleveland RB, Cleveland WS, McRae JE, Terpenning I (1990) STL: a seasonal-trend decomposition procedure based on loess (with discussion). J Off Stat 6:3–73

Cronholm S, Göbel H (2019) Evaluation of action design research. Scand J Inf Syst 31(2):2

Cronholm S, Göbel H (2022) Action design research – models for researcher-practitioner collaboration. In: DESRIST Int Conf Des Sci Res Inf Syst Technol, Springer Science and Business Media Deutschland GmbH, pp 393–404. https://doi.org/10.1007/978-3-031-06516-3_29

De Livera AM, Hyndman RJ, Snyder RD (2011) Forecasting time series with complex seasonal patterns using exponential smoothing. J Am Stat Assoc 106:1513–1527. https://doi.org/10.1198/jasa.2011.tm09771

Devanbu P, Zimmermann T, Bird C (2018) Belief and evidence: how software engineers form their opinions. IEEE Softw 35:72–76. https://doi.org/10.1109/MS.2018.4321246

Díaz O, Montalvillo L, Medeiros R, Azanza M, Fogdal T (2022) Visualizing the customization endeavor in product-based-evolving software product lines: a case of action design research. Empir Softw Eng 27(3):75

Eken B, Tufan S, Tunaboylu A, Guler T, Atar R, Tosun A (2021) Deployment of a change-level software defect prediction solution into an industrial setting. J Softw Evol Process 33:e2381. https://doi.org/10.1002/smr.2381

Elden M, Chisholm RF (1993) Emerging varieties of action research: introduction to the special issue. Hum Relations 46:121–142. https://doi.org/10.1177/001872679304600201

Figalist I, Elsner C, Bosch J, Olsson HH (2021) Fast and curious: a model for building efficient monitoring- and decision-making frameworks based on quantitative data. Inf Softw Technol 132:106458. https://doi.org/10.1016/j.infsof.2020.106458

Fuglede B, Topsoe F (2004) Jensen-Shannon divergence and Hilbert space embedding. Proceedings of the International Symposium on Information Theory. IEEE p 30. https://doi.org/10.1109/ISIT.2004.1365067

Galdi P, Tagliaferri R (2018) Data mining: Accuracy and error measures for classification and prediction. In: Encycl Bioinforma Comput Biol ABC Bioinforma. https://doi.org/10.1016/B978-0-12-809633-8.20474-3

Gregor S, Chandra Kruse L, Seidel S (2020) Research perspectives: the anatomy of a design principle. J Assoc Inf Syst 21(6):2

H2020 EU project - Quality Aware Rapid Software Development project. https://www.q-rapids.eu/. Accessed May 2024.

Haj-Bolouri A, Purao S, Rossi M, Bernhardsson L (2018) Action design research in practice: lessons and concerns. ECIS 131

Hyndman R, Koehler A, Ord K, Snyder R (2008) Forecasting with exponential smoothing. Springer, Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-71918-2

Hyndman R, Kostenko A (2007) Minimum sample size requirements for seasonal forecasting models, Foresight Int J Appl Forecast 6:12–15. https://ideas.repec.org/a/for/ijafaa/y2007i6p12-15.html. Accessed 24 Mar 2021

Hyndman RJ, Athanasopoulos G (2018) Forecasting: principles and practice, 2nd edn, OTexts, Melbourne, Australia. https://otexts.com/fpp2/. Accessed May 2024.

Hyndman RJ, Khandakar Y (2008) Automatic time series forecasting: the forecast package for R. J Stat Softw 27. https://doi.org/10.18637/jss.v027.i03

Hyndman RJ, Koehler AB (2006) Another look at measures of forecast accuracy. Int J Forecast 22:679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

Kumaresan K, Ganeshkumar P (2020) Software reliability prediction model with realistic assumption using time series (S)ARIMA model. J Ambient Intell Humaniz Comput 11:5561–5568. https://doi.org/10.1007/s12652-020-01912-4

M. Manzano, Time series gathering tool, (2021a). https://github.com/martimanzano/SSI-forecast-R_scripts/blob/IST-paper/TimeSeriesFunctions_GPL_R_elastic_1.R#L25. Accessed May 2024.

M. Manzano, Forecasting Execution Tool, (2021b). https://github.com/martimanzano/SSI-forecast-R_scripts/blob/IST-paper/TimeSeriesFunctions_GPL_R_elastic_1.R#L173 (R). https://github.com/

martimanzano/SSI-forecast/blob/IST-paper/src/main/java/Forecast/Elastic_RForecast.java#L480 (Java). Accessed May 2024.
M. Manzano, Distance computation tool, (2021c). https://github.com/martimanzano/SSI-assessment/blob/3d877924310473cf172959ac56efe352f8255c56/src/main/java/Util_Assessment_SI/BayesUtils.java#L171. Accessed May 2024.
M. Manzano, Model Comparison Tool, (2022a). https://github.com/martimanzano/SSI-forecast/blob/IST-paper/src/main/java/Forecast/Utils.java#L190. Accessed May 2024.
M. Manzano, Accuracy computation tool, (2022b). https://github.com/martimanzano/SSI-forecast/blob/IST-paper/src/main/java/Forecast/Utils.java#L359. Accessed May 2024.
M. Manzano, Forecasting report tool, (2022c). https://github.com/martimanzano/SSI-forecast/blob/IST-paper/src/main/java/Forecast/Utils.java#L395. Accessed May 2024.
M. Manzano, Autocorrelation Test Tool, (2021). https://github.com/martimanzano/SSI-forecast-R_scripts/blob/IST-paper/TimeSeriesFunctions_GPL_R_elastic_1.R#L539 (R). https://github.com/martimanzano/SSI-forecast/blob/IST-paper/src/main/java/Forecast/Elastic_RForecast.java#L279 (Java). Accessed May 2024.
Manzano M, Ayala C, Gómez C, Abherve A, Franch X, Mendes E (2021) A method to estimate software strategic indicators in software development: an industrial application. Inf Softw Technol 129:106433. https://doi.org/10.1016/j.infsof.2020.106433
M. Manzano, Model Fitting Tool, (2021). https://github.com/martimanzano/SSI-forecast-R_scripts/blob/IST-paper/TimeSeriesFunctions_GPL_R_elastic_1.R#L159 (R).https://github.com/martimanzano/SSI-forecast/blob/IST-paper/src/main/java/Forecast/Elastic_RForecast.java#L410 (Java). Accessed May 2024.
Marshall A, Mueck S, Shockley R (2015) How leading organizations use big data and analytics to innovate. Strateg Leadersh 43:32–39. https://doi.org/10.1108/SL-06-2015-0054
S. Martínez-Fernández, P. Jovanovic, X. Franch, A. Jedlitschka, Towards automated data integration in software analytics, in: BIRTE '18 Proc. Int. Work. Real-Time Bus. Intell. Anal., Association for Computing Machinery, 2018: pp. 1–5. DOI: https://doi.org/10.1145/3242153.3242159
S. Matsumoto, R. Carvalho, M. Ladeira, UnBBayes: a java framework for probabilistic models in AI, Http://Unbbayes.Sourceforge.Net/. (2011). http://sourceforge.net/projects/unbbayes/. (accessed February 13, 2020)
Mendes E, Rodriguez P, Freitas V, Baker S, Atoui MA (2018) Towards improving decision making and estimating the value of decisions in value-based software engineering: the VALUE framework. Softw Qual J 26:607–656. https://doi.org/10.1007/s11219-017-9360-z
Mikkonen T, Lassenius C, Männistö T, Oivo M, Järvinen J (2018) Continuous and collaborative technology transfer: software engineering research with real-time industry impact. Inf Softw Technol 95:34–45. https://doi.org/10.1016/J.INFSOF.2017.10.013
Misirli AT, Bener AB (2014) Bayesian networks for evidence-based decision-making in software engineering. IEEE Trans Softw Eng 40:533–554. https://doi.org/10.1109/TSE.2014.2321179
A. Mockus, Missing data in software engineering, in: Guid. to Adv. Empir. Softw. Eng., Springer London, 2008: pp. 185–200. DOI: https://doi.org/10.1007/978-1-84800-044-5_7
Newbold P (1983) ARIMA model building and the time series analysis approach to forecasting. J Forecast 2:23–35. https://doi.org/10.1002/for.3980020104
Oates BJ (2006) Researching information systems and computing. Sage Publications Ltd.
J Pearl (1985) Bayesian networks a model of self-activated memory for evidential reasoning. Proceeding 7th Conf Cogn Sci Soc 329–334
Pospieszny P, Czarnacka-Chrobot B, Kobylinski A (2018) An effective approach for software project effort and duration estimation with machine learning algorithms. J Syst Softw 137:184–196
Priya Varshini AG, Kumari AK (2020) Predictive analytics approaches for software effort estimation: a review. Indian J Sci Technol 13:2094–2103
Q-Rapids Dashboard Tool: https://github.com/q-rapids/q-rapids. Accessed May 2024.
Q-Rapids, Q-Rapids Deliverable D3.1, 2018. Available at: https://www.q-rapids.eu/_files/ugd/11d1c4_8f78d8a1a28e4ee492b605957f1c2d25.pdf. Accessed May 2024.
R. Development Core Team, R: A Language and Environment for Statistical Computing, Vienna Austria R Found. Stat. Comput. (2008)
Raja U, Hale DP, Hale JE (2009) Modeling software evolution defects: a time series approach. J Softw Maint Evol Res Pract 21:49–71. https://doi.org/10.1002/smr.398
Ralph P (2014) Lab-based action design research. ICSE Companion:528–531
P. Ralph, S. Baltes, D. Bianculli, Y. Dittrich, M. Felderer, R. Feldt, A. Filieri, C.A. Furia, D. Graziotin, P. He, R. Hoda, N. Juristo, B. Kitchenham, R. Robbes, D. Mendez, J. Molleri, D. Spinellis, M. Staron, K. Stol, D. Tamburri, M. Torchiano, C. Treude, B. Turhan, S. Vegas, ACM SIGSOFT Empirical Standards, (2020). https://arxiv.org/abs/2010.03525v2 (accessed February 28, 2022)

S. Raschka, Model evaluation, model selection, and algorithm selection in machine learning, 2018. http://arxiv.org/abs/1811.12808 (accessed January 24, 2020)
S. Reddivari and J. Raman, "Software Quality Prediction: An Investigation Based on Machine Learning," IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI), Los Angeles, CA, USA, 2019, pp. 115–122, doi: https://doi.org/10.1109/IRI.2019.00030
T. Rique et al."Shedding Light on the Techniques for Building Bayesian Networks in Software Engineering", in Anais do III Workshop Brasileiro de Engenharia de Software Inteligente, Campo Grande/MS, 2023, pp. 1–6, DOI: https://doi.org/10.5753/ise.2023.235744
Roumani Y, Nwankpa JK, Roumani YF (2015) Time series modeling of vulnerabilities. Comput Secur 51:32–40. https://doi.org/10.1016/j.cose.2015.03.003
Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng. https://doi.org/10.1007/s10664-008-9102-8
Schelter S, Biessmann F, Januschowski T, Salinas D, Seufert S, Szarvas G (2018) On Challenges in Machine Learning Model Management. IEEE Data Eng Bull 41:5–15
Sein MK, Henfridsson O, Purao S, Rossi M, Lindgren R (2011) Action design research, MIS Q. Manag Inf Syst 35:37–56. https://doi.org/10.2307/23043488
D. Shaub, P. Ellis, forecastHybrid: Convenient Functions for Ensemble Time Series Forecasts, (2020). https://github.com/ellisp/forecastHybrid. Accessed May 2024.
Song Q, Runeson P (2023) Industry-academia collaboration for realism in software engineering research: insights and recommendations. Inf Softw Technol 156:107135. https://doi.org/10.1016/J.INFSOF.2022.107135
Staron M, Meding W, Söderqvist B (2010) A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation. Inf Softw Technol 52(10):1069–1079
Tadeusiewicz R (1995) Neural networks: a comprehensive foundation. Control Eng Pract 3:746–747. https://doi.org/10.1016/0967-0661(95)90080-2
Taylor SJ, Letham B (2018) Forecasting at scale. Am Stat 72:37–45. https://doi.org/10.1080/00031305.2017.1380080
M. Thorström, Applying machine learning to key performance indicators, Chalmers University of Technology, 2017. https://odr.chalmers.se/handle/20.500.12380/250254 (accessed September 28, 2021)
Tosun A, Bener AB, Akbarinasaji S (2017) A systematic literature review on the applications of Bayesian networks to predict software quality. Softw Qual J 25:273–305. https://doi.org/10.1007/s11219-015-9297-z
Tosun A, McIntosh S, Minku L, Turhan B (2020) Guest editorial: special issue on predictive models and data analytics in software engineering. Empir Softw Eng 25:1533–1535. https://doi.org/10.1007/s10664-020-09811-0
Tsoukalas D, Kehagias D, Siavvas M, Chatzigeorgiou A (2020) Technical debt forecasting: an empirical study on open-source repositories. J Syst Softw 170:110777
Twala B, Cartwright M (2010) Ensemble missing data techniques for software effort prediction. Intell Data Anal 14:299–331. https://doi.org/10.3233/IDA-2010-0423
B. Twala, M. Cartwright, M. Shepperd, Comparison of various methods for handling incomplete data in software engineering databases, in: 2005 Int. Symp. Empir. Softw. Eng. ISESE 2005, 2005: pp. 105–114. DOI: https://doi.org/10.1109/ISESE.2005.1541819
Van Koten CC, Gray AR (2006) An application of Bayesian network for predicting object-oriented software maintainability. Inf Softw Technol 48(1):59–67
B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, V. Filkov, Quality and productivity outcomes relating to continuous integration in GitHub, in: Proc. 10th Jt. Meet. Found. Softw. Eng. ESEC/FSE '15, ACM Press, New York, New York, USA, 2015: pp. 805–816. DOI: https://doi.org/10.1145/2786805.2786850
S. Wagner: A Bayesian network approach to assess and predict software quality using activity-based quality models. In Proceedings of the 5th International Conference on Predictor Models in Software Engineering (pp. 1–9), 2009
Wieringa RJ (2014) Design science methodology for information systems and software engineering. Springer, Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-43839-8
Wohlin C, Runeson P (2021) Guiding the selection of research methodology in industry–academia collaboration in software engineering. Inf Softw Technol 140:106678. https://doi.org/10.1016/J.INFSOF.2021.106678
Wohlin C, Šmite D, Moe NB (2015) A general theory of software engineering: balancing human, social and organizational capitals. J Syst Softw 109:229–242. https://doi.org/10.1016/J.JSS.2015.08.009
W. Wu, W. Zhang, Y. Yang, Q. Wang, Time series analysis for bug number prediction, in: 2nd Int. Conf. Softw. Eng. Data Mining, SEDM 2010, 2010: pp. 589–596. https://ieeexplore.ieee.org/document/5542853. Accessed May 2024.
Yasasin E, Prester J, Wagner G, Schryen G (2020) Forecasting IT security vulnerabilities – an empirical analysis. Comput Secur 88:101610. https://doi.org/10.1016/j.cose.2019.101610

Claudia Ayala is an associate professor at the Universitat Politècnica de Catalunya (UPC) - BarcelonaTech. She received her PhD degree in Informatics from the same university. She was a postdoctoral ERCIM research fellow at the Norwegian University of Science and Technology (NTNU), Norway. She received the Most Influential Paper Award in IEEE RE 2022 and an ACM SIGSOFT and IEEE Computer Society TCSE recognition for her outstanding contributions to the Software Engineering community at ICSE 2022. She has acted as: Financial Chair of ESEM 2024, Virtualization co-Chair of ICSE 2022, Project Manager of ICSE 2021, Virtualization team of ICSE 2020, Program co-Chair for the Ibero-American Conference on Software Engineering (CIbSE) and several other scientific events such as Posters and Demos Track-RCIS 2017; Short papers and posters track-EASE 2015; Proceedings co-Chair- PROFES 2019 and CAiSE'12; Student Volunteer Chair-RE 2008. She is a regular reviewer of highly ranked journals such as IEEE Transactions on Software Engineering, Empirical Software Engineering Journal, and Information and Software Technology. Her current research interests include empirical software engineering, software architecture and quality, software engineering for machine learning, and open source software adoption.

Cristina Gómez is an associate professor at the Universitat Politècnica de Catalunya (UPC) - BarcelonaTech. She received her Ph.D. Degree in Informatics from the same university in 2003. She is a member of the GESSI and inSSIDE research groups. Her main interests are conceptual modelling, software engineering, software architectures, object-oriented design and software engineering for machine learning. She has taught extensively on these topics and also conducted research on them. More information at https://cristina-gomez.staff.upc.edu/.

Martí Manzano received his PhD degree in Computing from the Universitat Politècnica de Catalunya (UPC) - BarcelonaTech, Spain in 2023. He has participated in national and international research projects such as the H2020 Q-Rapids Project. His research interests include Artificial Intelligence, Biomedical Engineering, and Software Engineering, where he has published and served as a reviewer in leading journals such as IST and JSS.

Antonin Abherve is the Modelio Product Owner at Softeam with more than 18 years of experience. After his Master degree in Software Engineering at Paris' 6th University, he worked in the domains of Model Driven Engineering, Model Driven Architecture, Service Oriented Architectures, Enterprise Architecture, Business Processes, and Cloud Computing. He has participated in several French or EU research projects including Galaxy, REMICS, Q-Rapids and MODAClouds. He recently worked on migration of legacy applications to Cloud Computing platforms using a model driven engineering methodology. He was the lead developer of the ITEA 3 MEASURE project (https://itea3.org/project/measure.html).

Xavier Franch is professor at the Universitat Politècnica de Catalunya (UPC) - BarcelonaTech. He holds a PhD in Informatics. He is member of Academia Europea, AIIA Fellow, Vice-chair of the International Requirements Engineering Board (IREB) Council and member of the ISERN network. He has received the Life-Time Service Award of the IEEE RE conference and the Most Influential Paper Award in IEEE RE 2022 and REFSQ 2024. He is or has been: Editorial Board member at IST, JSS, REJ, Computing and IET Software journals; Editor-in-Chief at IET Software; J1 chair at JSS; Steering Committee member at IEEE RE, CAiSE, REFSQ and PROFES; PC chair at CAiSE'22, RE'16, ICSOC'14, CAiSE'12, REFSQ'11 and ICCBSS'06; General Chair at ESEIW/ESEM'24, REFSQ'23, RCIS'22, PROFES'19, RE'08.

Authors and Affiliations

Claudia Ayala[1] · **Cristina Gómez**[1] · **Martí Manzano** · **Antonin Abherve**[2] · **Xavier Franch**[1]

✉ Claudia Ayala
claudia.ayala@upc.edu

Cristina Gómez
cristina.gomez@upc.edu

Martí Manzano
mmanzano@essi.upc.edu

Antonin Abherve
antonin.abherve@softeam.fr

Xavier Franch
xavier.franch@upc.edu

1 Universitat Politècnica de Catalunya (UPC), C/Jordi Girona 1-3, Omega 117, 08034 Barcelona, Spain

2 Softeam Group, Paris, France