LLM-Based Automation of COSMIC Functional Size Measurement from Use Cases

Gabriele De Vito, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, Fabio Palomba

Abstract—COmmon Software Measurement International Consortium (COSMIC) Functional Size Measurement is a method widely used in the software industry to quantify user functionality and measure software size, which is crucial for estimating development effort, cost, and resource allocation. COSMIC measurement is a manual task that requires qualified professionals and effort. To support professionals in COSMIC measurement, we propose an automatic approach, CosMet, that leverages Large Language Models to measure software size starting from use cases specified in natural language. To evaluate the proposed approach, we developed a web tool that implements CosMet using GPT-4 and conducted two studies to assess the approach quantitatively and qualitatively. Initially, we experimented with CosMet on seven software systems, encompassing 123 use cases, and compared the generated results with the ground truth created by two certified professionals. Then, seven professional measurers evaluated the analysis achieved by CosMet and the extent to which the approach reduces the measurement time. The first study's results revealed that CosMet is highly effective in analyzing and measuring use cases. The second study highlighted that CosMet offers a transparent and interpretable analysis, allowing practitioners to understand how the measurement is derived and make necessary adjustments. Additionally, it reduces the manual measurement time by 60-80%.

Index Terms—Use Cases; Functional Size Measurement (FSM); COSMIC; Natural Language Processing (NLP); Large Language Models (LLMs)

1 Introduction

Measuring the size of a software system is crucial for supporting several software project management activities, including estimating development effort/cost, benchmarking, portfolio management, and assessing software contractor performance [1]–[6]. Several approaches exist for sizing software systems, such as Use Case Points [7], Story Points [8], object-oriented metrics [9]–[12], and Source Lines of Code (SLOC) [13]. Functional Size Measurement (FSM) methods have been introduced to measure software size based on functional requirements, making them independent of the technologies used to develop the software application. Function Point Analysis (FPA) was the first FSM method proposed [14]–[17]. Over time, several variants of FPA (e.g., MarkII and NESMA) have been developed to improve measurement accuracy or extend applicability to specific domains [18].

In 1999, COmmon Software Measurement International Consortium (COSMIC) FSM (hereafter COSMIC) [19] was introduced as a second-generation FSM, addressing the limitations of earlier methods like FPA in measuring real-time and embedded systems and dealing with modern software architectures. COSMIC has gained significant attention in recent years [20] due to its broader applicability across various domains and its ability to provide accurate effort predictions [18], [21]–[25]. COSMIC measures the size of Functional User Requirements (FURs), which describe what the software must do to satisfy users' needs. These requirements are mapped to the COSMIC Generic Software Model, which represents software functionality through data movements between functional users and persistent storage. Each data movement represents one COSMIC Function Point (CFP), providing a standardized measure of the functionality delivered to users. These data movements can be identified from various software artifacts such as requirements specifications, use cases, or user stories. UML use cases are considered one of the best approaches for requirements elicitation and specification because they represent interactions between the system and its environment as natural language text in a semi-formal way [26]–[28], making them particularly suitable for automated analysis. However, measuring COSMIC from textual requirements remains a manual process, demanding qualified professionals and considerable time. While significant research has been conducted to automate FSM methods, including COSMIC [29]–[49], there are still few studies specifically exploring the automation of COSMIC measurement starting from natural language requirements [28], [50]–[56]. Previous automation approaches have often relied on structured requirements or historical data and used text mining techniques [50], syntactic rule-based systems [52], or ontological analysis [51]. Recent significant advances in NLP stem from the introduction of deep neural networks and Large Language Models (LLMs), complex models consisting of a neural network trained on large quantities of unlabelled text via self-supervised learning [57]. By providing a suitable prompt, LLMs can generate accurate responses [58], [59]. Specific LLMs, such as Generative Pre-trained Transformer 4 (GPT-4) [60], Claude 3.5 [61], and LLama [62],

- *Gabriele, Fabio, Carmine, and Filomena are with the Software Engineering (SeSa) Lab of the University of Salerno, Italy. E-mails: {gadevito, fpalomba, gravino, fferrucci}@unisa.it*
- *Sergio is with the University of Naples Federico II, Italy. E-mail: sergio.dimartino@unina.it*

Manuscript received September XX, 2022; revised XX XX, XXXX.

have demonstrated remarkable capabilities for in-context learning, translation, and text generation [63] and several emergent abilities have been identified [64].

Given the impressive performance of these models, their potential in the context of COSMIC measurement is worth exploring. Still, in our experience (as detailed in Section 2.3), the direct application of current LLMs to perform COSMIC measurement based on use cases description is not providing useful results. LLMs struggle with tasks requiring deep logical reasoning, often producing incorrect outputs even if the input prompt is clear and unambiguous [65]. Additionally, their responses can vary unpredictably with different prompts [66], which is problematic for the consistency required in FSM. Nonetheless, we conjecture that it is possible to develop LLM-based solutions to automate the application of the COSMIC method to use cases, with the dual purpose of measuring the functional size and providing a transparent and detailed analysis of the COSMIC concepts applied to the functional requirements. To this aim, we introduce CosMet (COSMIC MEasurementT), an approach to automate COSMIC measurement based on use case specifications, based on LLMs. In particular, CosMet takes as input the textual descriptions of use cases, including actors, their main scenarios, alternative flows, and exceptions, and produces the complete COSMIC analysis report as output. The approach includes two main components: (1) the Sentence Splitter Component and (2) the COSMIC Analyzer Component. The Sentence Splitter is designed to split complex sentences into use-case scenarios, breaking them into constituent steps, each representing a single atomic action. Then, the COSMIC Analyzer Model processes the resulting use cases to produce a COSMIC measurement, identifying Functional users, Triggering events, Objects of interest, Data Groups, and Data movement types, and then calculating the COSMIC Function Points. Both of these components are intended as refined instances of LLMs, using a few-shot learning. To assess CosMet, we developed a web tool based on GPT-4 and performed an empirical study on seven software systems, specified by 123 use cases, from Management Information Systems, Telemedicine, Real-Time systems, and ML applications representing real-world software projects from both industry and reference case studies. We evaluated the results from both quantitative and qualitative perspectives. Our investigation revealed that CosMet effectively identified data movements, achieving a 99% F1 score for both Rouge and BLEU metrics. The tool correctly mapped functional requirements to the COSMIC Generic Software Model, with Rouge and BERTscore F1 scores ranging from 84.5% to 100%. Compared to manual measurement, it reduced measurement time by 60-80%.

The main contributions of the paper are fourfold:

1) We introduce a novel LLM-based approach, CosMet, which aims to measure use cases and map Functional User Requirements to the COSMIC Generic Software Model.
2) We develop an automated tool that implements the CosMet approach using GPT-4.
3) We provide datasets, including use case specifications and their measurement, as a contribution to the research community.
4) We evaluate CosMet's effectiveness in measuring swoftware systems starting from use case specifications, mapping FURs to the COSMIC Generic Software Model, and reducing the measurement time. Furthermore, we systematically compare CosMet with existing COSMIC automated approaches to highlight its relative strengths and weaknesses.

These contributions are significant as they can serve as a considerable milestone and the foundation for future refinement of the application of LLMs in the FSM field, such as extending the approach to FPA and comparing COSMIC with different FSM methods. This study focuses explicitly on COSMIC measurement automation, as comparing COSMIC with other methods like FPA would introduce variables related to the inherent differences between FSM methods that are beyond the scope of this work.

With our study, we hope to contribute to notably improve functional size measurement in the software industry, making it more accurate, efficient, and cost-effective.

Structure of the paper. Section 2 provides the background and motivation of our work. Section 3 presents CosMet, while the research method employed to assess it is described in Section 4. Section 5 reports the results, while the main findings and the discussion of threats to validity are provided in Section 6. Section 7 covers the related work. Section 8 concludes the paper and outlines future work directions.

2 BACKGROUND

In the following, we provide the necessary background to comprehend our work and its relevant context. First, we briefly introduce COSMIC. Next, we explore the role of LLMs, which form the core components of the proposed approach, and discuss the motivation driving our research.

2.1 COSMIC

The first Functional Size Measurement method was FPA, introduced in the late 1970s [14]–[17]. It measures software size by counting functional components such as internal logical files, external interface files, external inputs, external outputs, and external inquiries, with each component weighted according to its complexity. While FPA proved valuable for traditional information systems, it revealed limitations when applied to modern software architectures and real-time systems. To address these challenges and accommodate a broader range of application domains, COSMIC was proposed in 1999 as a second-generation FSM, offering significant advancements over FPA in the measurement process. The adoption of COSMIC as an industrial standard continues to grow[1], largely due to the organization's global presence, with representatives in 26 countries and numerous companies and institutions publicly recognizing their use of the COSMIC method. COSMIC provides a comprehensive framework of rules, concepts, and processes specifically tailored for measuring the functional size of software based on its FURs [19]. FURs capture interactions between a Functional User, who acts as both a sender and receiver of data, across the Boundary and with Persistent Storage within this Boundary. The software architecture is structured into

[1]COSMIC. World-wide usage. https://cosmic-sizing.org/usage/

Layers, and the software component being measured must belong to a single layer. The measurement process is guided by a clearly defined Purpose, which explains the importance of the measurement, and a well-specified Scope, identifying the specific FURs to be measured.

To measure software, the FURs must align with COSMIC's Generic Software Model, which provides the necessary concepts and principles for this aim. Within this model, the measurement of FURs is guided by several key concepts. A Functional process represents a set of FURs that form a logical grouping of data movements and can be executed independently. It is initiated by a Triggering event from a Functional user and consists of a sequence of sub-processes. Data movements within these processes are referred to as Functional sub-processes. A Data group is defined as a set of data attributes that characterize a unique aspect of an object of interest, representing the smallest meaningful unit of information within the data group. The functional size of software is determined by identifying and counting the data movements within each functional process. These movements include: Entries (E), where data groups are transferred from a Functional user into the process; Exits (X), where data groups are sent from the process to the Functional user; Reads (R), where data groups are moved from persistent storage to the process; and Writes (W), where data groups are transferred from the process to persistent storage. Each data movement is counted as one COSMIC Function Point (CFP). Calculating the size of a software component within a defined scope involves summing up the sizes of all identified Functional processes. Each process must include at least two CFPs, typically formed by an E plus either a W or an X.

Use Case Example: Insert a New Customer

Description: This function lets the user add a new customer.
Primary actor: Hotel manager
MAIN Scenario

1) *The user clicks the "New customer" button.*
2) *The system shows a form with the following editable fields: surname, name, date of birth, fiscal code, address, city, and notes.*
3) *The user fills in the surname, name, date of birth, address, city, email address, fiscal code fields, and optionally the notes field and submits the form.*
4) *The system checks that: the surname and name fields contain at least two characters, the fiscal code field is valid, the date of birth is correct, the email address is valid and records the new customer in persistent storage.*
5) *The system shows a confirmation message.*

Exceptions
5.a1 The system shows an error message stating that the provided data are not valid.

Fig. 1: Use Case: Insert a New Customer

To illustrate the COSMIC measurement concepts, consider the use case in Figure 1 taken from a hotel management system. Tn this scenario, a hotel manager (the functional user) enters a new customer's details into the system. Table 1 demonstrates the application of the COSMIC method to the specifications outlined in this use case. The functional size is calculated as 3 CFP, derived from three distinct data movements:

TABLE 1: COSMIC analysis of the "Insert new customer" use case

Triggering Event: User clicks "New customer" button

FU	Sub-process	DG	OOI	DM	CFP
Hotel manager	Click new customer button	-	-	-	
	Display customer form				
Hotel manager	Enter customer data	Customer data	Customer	E	1
	Validate and store customer	Customer data	Customer	W	1
	Show confirmation / errors	Message	Message	X	1
				Total:	**3**

FU: Functional User, DG: Data Group
OOI: Object of Interest, DM: Data Movement Type

an Entry (E) when the user submits the customer data, a Write (W) when the system stores the validated data, and an Exit (X) when the system displays either a confirmation or error message. when the system displays either a confirmation or error message. Each of these movement involves the "Customer" data group, which is associated with the "Customer" object of interest.

Despite its effectiveness, applying COSMIC to analyze such processes is time-consuming and necessitates the involvement of professional experts for accurate measurement. Consequently, a key research objective within the COSMIC community is to automate the measurement process to facilitate broader industrial adoption [51].

2.2 Large Language Models

The NLP field has experienced a significant surge in interest thanks to LLMs, which are deep neural networks with numerous parameters trained on extensive text corpora. LLMs produce human-like writing and excel in various language-related tasks, such as translation and summarization [67]–[69]. The initial training process, known as "pre-training", is computationally demanding and time-consuming due to the extensive datasets required to train the models. LLMs can later be specialized through a fine-tuning process, using smaller datasets tailored for executing new NLP tasks (e.g., named-entity recognition, sentiment analysis) or performing the same tasks in different domains. From an architectural point of view, these models are based on the Transformer and the self-attention mechanisms introduced by Vaswani et al. [70] for predicting the next-word in a phrase. The Transformer architecture consists of encoders and decoders [71]. Encoders transform input words into vector representations, refined by self-attention to predict subsequent words. Decoders then use these vectors to determine the probability of potential subsequent words. Presently, many LLMs exist, ranging from commercial services like GPT-3 [71] and GPT-4 [60] to open-source alternatives like BERT [72], FLAN-T5 [73], and LLama [62], each offering distinct capabilities and limitations. GPT-4 is a powerful option surpassing many predecessors [2] and current systems across various NLP benchmarks [60], [74], and it was the model we employed in our experiments.

[2]Trustbit. *LLM Leaderboard - July 2024*. [Online]. Available: https://www.trustbit.tech/en/llm-leaderboard-juli-2024.

However, our approach is not exclusively bound to any single LLM, allowing for flexibility to choose the most suitable model for parsing language in use case specifications and producing COSMIC measurement. LLMs like GPT-4 incorporate "in-context learning" capabilities, enabling the model to generalize to new scenarios with minimal to no examples. These models possess different context window size, determining the number of tokens that can be exchanged in a single interaction between the user and the LLM. GPT-4 can accommodate 8,192 tokens, enabling the processing of extensive text demonstrations within a single interaction. To effectively use LLMs, researchers and practitioners employ prompt engineering techniques to guide the model's responses [75], [76]. One common approach is few-shot learning [77], where the model learns from a small number of examples. Several studies [78]–[82] have shown that LLMs can effectively adapt to new tasks across different domains using just a few examples. These characteristics make LLMs particularly suitable for COSMIC measurement because (1) they can understand and process natural language descriptions in use cases, (2) they can learn to identify COSMIC concepts (like functional users and data movements) through few-shot examples, and (3) their context windows allow them to analyze entire use cases while maintaining consistency in the measurement process.

2.3 Motivation of our work

While LLMs show great potential for processing natural language requirements, their effective application to COSMIC measurement requires a structured approach.

Before developing CosMet, we indeed verified GPT-4's baseline knowledge of COSMIC by directly asking it to explain the COSMIC FSM method and its steps, as shown in figure 2. The model demonstrated a solid understanding of the core concepts by correctly describing the fundamental purpose of COSMIC, the concept of functional processes, the four types of data movements (Entry, Exit, Read, and Write), and the six main steps of the method, ranging from scope identification to final size calculation. However, while this baseline knowledge was encouraging, GPT-4 cannot apply the methodology correctly. To demonstrate this, let us consider examples of direct application of GPT-3.5 and GPT-4 to COSMIC measurement, as detailed in Figure 3.

Upon reviewing the response from GPT-3.5, reported in Figure 4, it is evident that the COSMIC analysis requires adjustments. Indeed, GPT-3.5 missed the Object of Interest "Message". In addition, it made the following mistakes:

- Step 2 does not contain any data movement.
- Step 3 is not a "Read" data movement, but it is an "Entry" data movement.
- Steps 5 and 5.a1: They are "Exit" data movements but must be counted once.

Similarly, GPT-4's analysis (shown in Figure 5) also needs corrections because it missed the Object of Interest "Message" and made the following mistakes:

- Steps 1,2, and 4 do not contain any data movement.
- The total number of CFPs is not 7, but 3.

GPT-4 COSMIC Knowledge.

Prompt:
Please briefly explain what the COSMIC Functional Size Measurement method is and what the method steps are.

GPT-4 answer:
The COSMIC method is a standardized approach for quantifying software functionality, independent of implementation technology. It aids in estimating development effort, comparing productivity, and managing projects. The method revolves around "functional processes," defined as data movements triggered by meaningful user events, focusing on four types: Entry, Exit, Read, and Write. Here are the typical steps involved in the COSMIC FSM method:

1) *Define what part of the software will be measured and why the measurement is being done.*
2) *Determine the functional user requirements (FURs) that the software must fulfill.*
3) *Create a model representing the software's functionalities.*
4) *For each functional process, identify data movements. Classify each data movement as an Entry (E), Exit (X), Read (R), or Write (W).*
5) *Count each identified data movement type (E, X, R, W) as one "COSMIC Function Point" (CFP).*
6) *Sum up the CFPs from all data movements to get the total functional size of the software.*

Fig. 2: GPT-4 COSMIC Knowledge.

Example: Prompt to analyze and measure a use case

Act as an expert COSMIC measurer.

USE CASE:
UC-GES-03.4 Insert a new customer
Description: This function allows the user to insert a new customer.
Primary actor: Hotel manager, Operator
MAIN Scenario

1) *The user clicks the "New customer" button*
2) *The system shows a form with the following editable fields: surname, name, date of birth, fiscal code, address, city, and notes.*
3) *The user fills in the surname, name, date of birth, address, city, email address, fiscal code fields, and optionally the notes field and submits the form.*
4) *The system checks that: the surname and name fields contain at least two characters, the fiscal code field is valid, the date of birth is correct, the email address is valid and records the new customer in persistent storage.*
5) *The system shows a confirmation message.*

Exceptions
5.a1 The system shows an error message stating that the provided data are not valid.

Analyze the use case and report a detailed COSMIC Functional Size Measurement analysis, providing the Triggering event for the functional process and the functional users, Data Groups, Objects of interest, Data movement types, and CFP for each use case step.

Fig. 3: Prompt to analyze and measure a use case.

GPT-3.5 COSMIC analysis

Triggering event: The user clicks the "New customer" button
Functional Users: Hotel manager, Operator
Data Groups: Customer Information
Objects of Interest: Customer
Data Movement Types:

1) *Read: The system reads the existing customer data from persistent storage.*
2) *Write: The system writes the new customer data to persistent storage.*

CFP for each use case step:

1) *The user clicks the "New customer" button - 0 CFP*
2) *The system shows a form with the following editable fields: surname, name, date of birth, fiscal code, address, city, and notes. - 1 CFP (Read)*
3) *The user fills in the surname, name, date of birth, address, city, email address, fiscal code fields, and optionally the notes field and submits the form. - 1 CFP (Read)*
4) *The system checks that: the surname and name fields contain at least two characters, the fiscal code field is valid, the date of birth is correct, the email address is valid and records the new customer in persistent storage. - 1 CFP (Write)*
5) *The system shows a confirmation message. - 0 CFP*

Exceptions:
5.a1 The system shows an error message stating that the provided data are not valid. - 0 CFP
Total CFP for the use case: 3 CFP

Fig. 4: GPT-3.5 COSMIC analysis.

GPT-4 COSMIC analysis

Triggering Event: The user clicks the "New customer" button.
Functional Users: Hotel Manager, Operator
Data Groups and Objects of Interest:

1) *Customer Details (Surname, Name, Date of Birth, Fiscal Code, Address, City, Email Address, Notes)*

Data Movement Types:

1) *Entry: The user fills in the customer details and submits the form.*
2) *Exit: The system shows a confirmation message.*
3) *Read: The system checks the entered customer details.*
4) *Write: The system records the new customer in persistent storage.*

COSMIC Functional Size Measurement (CFP) Analysis:
Step 1: The user clicks the "New customer" button.
- Triggering event: External Input (EI), CFP: 1
Step 2: The system shows a form with editable fields.
- Data movement: External Output (EO), CFP: 1
Step 3: The user fills in the customer details and submits the form.
- Data movement: External Input (EI), CFP: 1
Step 4: The system checks the entered customer details.
- Data movement: External Inquiry (EQ), CFP: 1
Step 4: The system records the new customer in persistent storage.
- Data movement: External Input (EI), CFP: 1
Step 5: The system shows a confirmation message.
- Data movement: External Output (EO), CFP: 1
Exception 5.a1: The system shows an error message stating that the provided data are not valid.
- Data movement: External Output (EO), CFP: 1

Total CFP: 7

Fig. 5: GPT-4 COSMIC analysis.

Moreover, it used an incorrect vocabulary. In particular, it reported "External Input", "External Inquiry", and "External Output", which are not "Base Functional Component" types in the COSMIC method. Therefore, there is a need for a more accurate and reliable method to analyze and measure software systems starting from use case specification using LLMs. This motivates our work.

3 The CosMet Approach

The CosMet approach comprises two components (see Figure 6), namely (1) the *Sentence Splitter* and (2) the *COSMIC Analyzer*, intended as two refined instances of GPT-4, to analyze use cases specified in natural language. First, the Sentence Splitter examines the use case scenarios and divides complex sentences into single steps, each representing an atomic action performed by the system or the user, ensuring that each action can be distinctly measured and analyzed for accurate COSMIC functional size measurement. Then, the COSMIC Analyzer applies the COSMIC method to these atomic actions, identifying Functional users, Triggering events, Objects of interest, Data groups, and Data movement types, and calculating the number of CFPs.

For example, given the 'Insert a New Customer' use case presented in Section 2.1, CosMet produces an analysis report as shown in Table 1. To implement the two components, we employed GPT-4 as LLM and followed four main steps (described in detail in the following): (1) prompt engineering, (2) examples preparation, (3) validation, and (4) hyperparameters tuning. Additionally, we developed a web application that interacts with the GPT-4 API.

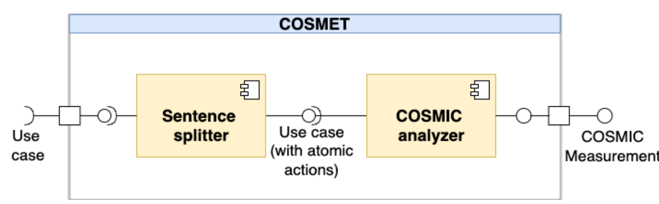

Fig. 6: CosMet components

3.1 Prompt Engineering

We employed the few-shot learning pattern [77] for the implementation of the Sentence Splitter and COSMIC Analyzer components. As described in Section 2.2, this pattern leverages LLMs' ability to identify and learn patterns from a limited set of examples and then apply these patterns to new, similar cases. By providing well-crafted examples, LLMs can recognize the underlying structure and relationships in the data, enabling them to perform complex tasks without extensive training.

For the Sentence Splitter, few-shot examples were designed to teach the model how to identify complex sentences containing multiple actions and break them down into atomic steps. For example, when provided with examples of how to split sentences such as “The system validates the data and stores it in the database” into separate actions, the model learns to recognize similar patterns in new use cases. Similarly, for COSMIC analysis, few-shot examples illustrated how to identify key COSMIC concepts within use case descriptions. By presenting the model with examples that demonstrate the recognition of data movements, functional users, and objects of interest across various contexts, it learns the patterns that characterize these elements.

However, obtaining the desired outcomes with few-shot learning could be challenging. Human users must rely on a combination of domain expertise and creativity, experimenting with different prompts to achieve the desired results. To effectively address the challenges related to prompt engineering, we followed three main guidelines [83], [84]:

1) Show and tell: making intentions clear using examples, instructions, or both.
2) Provide quality data: ensuring enough samples for the model to follow a pattern.
3) Check settings: setting the model parameters to make the model predictable in responding.

Considering these guidelines, we prepared the prompt templates (as shown in Figures 7 and 8) and the training set to refine GPT-4 with few-shot examples using OpenAI Playground (a web-based interface offered by OpenAI), as described in the next section.

> **Sentence Splitter: Prompt Templates.**
>
> ***System Prompt:***
> *Split all the following sentences, numbering the new sentences so that each new sentence is grammatically correct.*
>
> *Example 1:*
> *...*
>
> ***User Prompt:***
> *{uc}*
> *Split:*

Fig. 7: Prompt templates for the Sentence Splitter.

For the sake of simplicity, Figures 7 and 8 present only the prompt templates. Specifically, the Sentence Splitter template follows a simple structure where the system prompt instructs the model to split complex sentences while maintaining grammatical correctness, followed by examples. The user prompt then presents the use case to be processed. The COSMIC Analyzer template is more structured, showing how to format the input use case and the expected measurement output. The system prompt includes examples of functional processes with their complete COSMIC analysis (triggering events, functional users, and sub-processes with data movements). In contrast, the user prompt presents the new use case to be analyzed following the same structure. Both templates are designed to guide the model through few-shot learning while maintaining consistency in the output format.

> **COSMIC Analyzer: Prompt Templates.**
>
> ***System Prompt:***
> *FP ″{Functional process 1}″:*
> *use case text...*
>
> *FP ″{Functional process 1}″ measurement:*
> *Triggering Event: {Triggering event description}*
> *Functional User: {Functional User}*
> *Sub-Processes:*
> *1. step description...*
> *FU (user) – DG (Data group) – OOI (Object of interest) – Data movement*
> *2. step description...*
> *...other examples...*
>
> ***User Prompt:***
> *FP ″uc_name″:*
> *{uc}*
>
> *FP: ″{uc_name}″ measurement:*
>
> *Do not add any explanation or anything else.*

Fig. 8: Prompt templates for the COSMIC Analyzer.

3.2 Examples Preparation

The training set preparation is the crucial step of every Machine Learning (ML) approach. In this stage, we addressed three critical issues. First, use cases can be written using different linguistic styles. Software engineers may group actions (executed by the system or the user) in use case scenarios. For instance, consider the following sentence:

“The system locates the reservation header and, if exists, details regarding the booked rooms and services, and the children's ages”.

In the example above, the system executes four actions:

1) Read (reservation header): The system searches for the reservation header.
2) Read (booked rooms): The system searches for the booked rooms.
3) Read (requested services): The system searches for the requested services.
4) Read (children's ages): The system searches for the children's ages.

The COSMIC method must identify the atomic data movement types to count them accurately. Despite GPT-4 being capable of few-shot learning on diverse tasks, it cannot identify all the possible combinations of data movements present in use case scenarios, as illustrated in Figures 9 and 10. Indeed, without splitting the sentences and providing the prompt in Figure 9, GPT-4 wrongly identifies three Read data movements and an Entry. On the other hand, using the prompt in Figure 10 in which the sentences are split, it correctly identifies the four Read data movements but erroneously pinpoints an Entry. Therefore, making actions atomic in the flow of events is essential.

GPT-4 Data Movements Identification without splitting.

Prompt:
You are an expert COSMIC FSM practitioner. Your task is to identify all the data movement types in the use case I'll provide, and the total number of CFP.

Use case:
The system locates the reservation header and, if exists, details regarding the booked rooms and services, and the children's ages.

Data movements:
Total CFP:
Do not add anything else.

GPT-4 answer:
Data movements: Entry (E), Read (R), Read (R), Read (R)
Total CFP: 4

Fig. 9: GPT-4 Data Movements Identification without splitting.

GPT-4 Data Movements Identification with splitting.

Prompt:
You are an expert COSMIC FSM practitioner. Your task is to identify all the data movement types in the use case I'll provide, and the total number of CFP.

Use case:
The system locates the reservation header.
If the reservation header exists, the system locates details regarding the booked rooms.
If the reservation header exists, the system locates details regarding the booked services.
If the reservation header exists, the system locates the children's ages.

Data movements:
Total CFP:
Do not add anything else.

GPT-4 answer:
Data movements: Entry (E), Read (R), Read (R), Read (R), Read (R)
Total CFP: 5

Fig. 10: GPT-4 Data Movements Identification with splitting.

The second issue concerns mapping the FURs expressed in the use cases to COSMIC's Generic Software Model (GSM) and applying COSMIC's rules. This is the hardest part of our prompt engineering because it is not a typical task for GPT-4 (i.e., classification, summarization, etc.). For example, according to the COSMIC manual, errors and confirmation messages must be classified as Exit, but they must be counted only once per Functional process. So, these kinds of data movements must be differentiated from the others to avoid inconsistent measurements. In addition, when the system exchanges data movements with external components, these must be classified as Exit (from the system towards the external components) or Entry (from the external components to the system).

The last challenge is related to the limited context size of GPT-4, namely the total length of the input message and the model response. The maximum context size of GPT-4 is 8,192 tokens. Therefore, it is critical to balance the prompt and the response size.

To efficiently manage these constraints, we refined GPT-4 using two training sets of few-shot examples (implemented in the prompt.py script under the folder CosMet/cosmic in our online repository [85]): T1 for the Sentence Splitter Component and T2 for the COSMIC Analyzer Component. The preparation of both training sets was carried out collaboratively by the authors, combining the practical industry knowledge of the first author (with over 30 years of experience in the IT field and more than 18 years in FSM) and the expertise of experienced researchers (with research experience ranging from 15 to 35 years).

These training sets are implemented as system prompts containing instructions and examples, following the structure described in Section 3.1. We carefully managed the token allocation during refinement to balance the few-shot examples and the space required for the model's output. We left over 5,200 tokens to process a use case and produce the COMSIC analysis. The number of tokens was strategically determined to be ample enough for verbose use cases, allowing for a comprehensive analysis without exceeding GPT-4's context size.

3.2.1 Sentence Splitter Training Set

The Sentence Splitter's prompt engineering involved creating clear instructions to help GPT-4 split complex sentences. We manually created a set of examples, drawing from our experience in software requirements engineering and COSMIC measurement. Additionally, we analyzed other use case documents and user stories from the companies that provided us with the evaluation and test datasets, which helped us understand their typical linguistic patterns.

We refined these prompts iteratively, focusing on preventing over-splitting sentences and preserving the semantic integrity of the use case steps. We used examples of varying complexity from different software fields, ensuring linguistic diversity and the ability to handle domain-specific terminology. This approach exposed the LLM to various sentence structures and action sequences from real-world use case documents.

As a result, the training set T1 (publicly available in the CosMet online repository [85] in the prompt.py script under the folder CosMet/cosmic) consisted of a context instruction and 16 example sentences followed by the expected result.

T1 was designed to split sentences taking into account the following semantic rules:

- If the sentence elaborates on an action without indicating data exchange between user, system, storage, or external components, it should remain unsplit (rule 1).
- If the sentence involves multiple actions by the same subject, each action must be listed as a separate use-case step, with the subject restated for clarity (rule 2). This often happens when the action has different verbs or periods.
- If the sentence regards a transaction and reports a validation status before (or after) that, then the sentence must not be split (rule 3).
- If a sentence describes a transaction involving multiple entities, it must be split (rule 4). This applies when multiple entity names follow the transaction verb, such as "The system inserts the reservation and the booked rooms," involving both "reservation" and "booked rooms."
- If the sentence reports a system message and explains its meaning, it must not be split (rule 5).

Figure 11 shows the response of the Sentence Splitter Component for the following use case example:

> **Example: Use case to split**
>
> *MAIN Scenario*
> *1) The hotel manager clicks the "Services" Link.*
> *2) The system shows a form to search services by code.*
> *3) The user optionally fills in the service code to filter the results and executes the search.*
> *4) The system searches for services that meet the search criteria and displays the list of available services.*
> *Exceptions:*
> *The system shows an error message stating that no results match the search criteria.*

MAIN SCENARIO:
1. The hotel manager clicks the "Services" link.
2. The system shows a form to search services by code.
3. The user optionally fills in the service code to filter the results and executes the search.
4. The system searches for services that meet the search criteria.
5. The system displays the list of available services.

Exceptions:
6. The system shows an error message stating that no results match the search criteria.

Fig. 11: GPT-4 response for a typical "split sentences" request

In the response, step no. 4 is split into two distinct steps: one for the search and the other for displaying the results. This splitting is necessary as each action corresponds to a different data movement (Read for searching and Exit for displaying). In contrast, step 3 is not split because, while it contains two actions (filling in the form and executing the search), only the search execution represents a meaningful data movement (Entry). This selective splitting approach helps optimize token usage and processing efficiency while ensuring accurate COSMIC measurement.

3.2.2 COSMIC Analyzer Training Set

We ensured the quality of the few-shot examples through a meticulous review process for the COSMIC Analyzer Component. The first author prepared each example following COSMIC measurement principles. The co-authors then reviewed these examples to ensure correctness and alignment with COSMIC standards. Consequently, the training set T2 (publicly available in our online repository [85]) comprised 7 examples of use cases with their relative COSMIC analysis: 3 examples coming from the Course Registration System (C-REG) case study [86] ("Enquire on a Professor's details", "Add a Professor's details", and "Delete a Professor's details"), and 4 examples engineered to teach GPT-4 how to deal with batch processes and typical message exchanges with external components in different domains ("Search for ships", "Reserve cabins", "Activate sensors", and "Activate processes").

These examples were crafted based on our experience with COSMIC measurement, following the same approach used for the Sentence Splitter training set. Figure 12 shows the measurement response with T2 for the following request:

> **Example: Use case measurement**
>
> *FP: "Search for rooms":*
> *MAIN SCENARIO:*
> *1) The hotel manager selects the sub-option "Rooms"*
> *2) The system shows a form to search rooms by code*
> *3) The user optionally fills in the room code to filter the results and executes the search*
> *4) The system searches for rooms that meet the search criteria*
> *5) The system displays the list of rooms that meet the search criteria*
> *6) The Hotel Manager browses the results*
> *Exceptions:*
> *7) The system shows an error message stating that no results match the search criteria.*
> *FP: "Search for rooms" measurement:*

FP: "Search for rooms" measurement:
Triggering Event: The Hotel Manager selects the sub-option "Rooms"
Functional User: Hotel Manager
Sub-Processes:
MAIN SCENARIO:
1. The hotel manager selects the sub-option "Rooms"
   No data movement.
2. The system shows a form to search rooms by code
   No data movement.
3. The user optionally fills in the room code to filter the results and executes the search
   FU (Hotel Manager) - DG (Room code) - OOI (Room) - Entry.
4. The system searches for rooms that meet the search criteria
   DG (Room code) - OOI (Room) - Read.
5. The system displays the list of rooms that meet the search criteria
   DG (Room details) - OOI (Room) - Exit.
6. The Hotel Manager browses the results.
   No data movement.

Exceptions:
7. The system shows an error message stating that no results match the search criteria.
   DG (Messages) - OOI (Message) - Exit. It must be counted only once.

Fig. 12: GPT-4 response for the "Functional process measurement" request

Following the same structure introduced in Section 2.1 (see Table 1), the output includes the Triggering Event that initiates the functional process and the Functional User involved. For each step in the scenario, the analysis identifies Data Groups (DG) and their Objects of Interest (OOI), specifies Data Movement types (Entry, Exit, Read), and indicates cases where no data movement occurs. The output also includes special annotations for Exits that should be counted only once, such as error messages.

3.3 Validation Process

The outputs produced by both components were carefully validated before being used as references for the approach. This validation process combined multiple sources of expertise and real-world examples. Initially, the first author reviewed the outputs and certified them in COSMIC measurement to ensure compliance with COSMIC principles and industry practices. Then, they underwent cross-checking with the other authors to ensure theoretical correctness. Additionally, we compared our results with actual use cases and measurements provided by the companies who supplied our evaluation datasets. While these additional materials cannot be publicly shared due to confidentiality agreements, they helped ensure that our examples and their analysis reflected real-world practices and measurement standards.

In addition, before conducting the complete empirical evaluation, we prepared two preliminary validation sets to verify the effectiveness of our few-shot examples (publicly available in our online repository [85]). The first set was used to validate the Sentence Splitter Component (V1), comprising 20 sentences extracted from real use cases provided by our industry partners. These sentences were specifically selected because they represent complex scenarios commonly found in industry use cases, such as multiple actions with different subjects, nested conditions, and compound transactions. This complexity helped us verify that our splitting approach could handle the various writing styles and structures typically found in real-world requirements. The second set validated the COSMIC Analyzer Component (V2). It was derived from the use case specifications of a Telemedicine microservice application, FIDDIA, developed by an Italian Small Medium Enterprise (Kiranet). The first author analyzed and measured these specifications, and the results were reviewed and validated by the co-authors. The use case specifications include four use cases and four Functional processes, providing a realistic test case for our measurement approach. Both validation sets were distinct from the training and test sets used in the empirical evaluation (see Section 4).

3.4 Hyperparameters tuning

Hyperparameter tuning is vital to get the best performance from LLMs and make them more predictable in responding. GPT-4, on which V1 and V2 components rely, has several parameters that allow us to adjust it:

- **Temperature:** The temperature parameter governs the model's predictability and originality. Its values range from 0 to 1. A temperature close to 1 results in more randomness and diversity, while a temperature near 0 makes the model's output more deterministic and predictable. Temperature values between 0.5 and 0.9 are commonly chosen for creative tasks to balance coherence and originality.
- **Top_p:** top_p sampling (nucleus sampling [87]) is used in text generation with LLMs to control the diversity of generated content. It sets a threshold to include only the most probable words that cumulatively exceed a specified probability "p.' Tuning top_p allows for a balance between creativity and coherence in the model's output. The OpenAI manual documentation[3] suggests utilizing one or the other option and setting the unused parameter to the default value.
- **Best_of:** It enables the generation of multiple completions server-side, with the 'best' response being the one with the lowest logarithmic probability per token.
- **Frequency_penalty:** The frequency penalty parameter controls the model's proclivity to make repeated predictions. The frequency penalty diminishes the likelihood of previously created words. The penalty is determined by how many times a word appears in the prediction.
- **Presence_penalty:** It encourages the model to create novel predictions by the presence penalty parameter. The presence penalty reduces the likelihood of a word if it previously was in the predicted text. Unlike the frequency penalty, the presence penalty is unaffected by the frequency with which terms appear in previous predictions.

To test and tune the Sentence Splitter and COSMIC Analyzer components, we used the GridSearch algorithm with Cross-Validation from Scikit learn library [88]. The algorithm exhaustively searches a subset of hyperparameter values for the refined components and uses a stratified cross-validation technique to split the data into subsets of folds. This ensures that each fold has a proportional representation of the target variable, making the evaluation more reliable. For each combination of hyperparameters, the algorithm splits the validation sets into four folds, then trains on three folds while testing on the remaining one. This process is repeated four times, with each fold serving as the test set once, and the average F1-score is computed across all folds. This cross-validation approach ensures robust evaluation of each hyperparameter configuration and helps prevent overfitting to specific examples.

We provided only a subset of the GPT-4 hyperparameters to the algorithm, to reduce the search space, making the following assumptions:

- **Best_of** must be set to 1 (the default value) because we need only the 'best' response server-side.
- **Top_p** must be set to 1 (the default value) because we employed the temperature in the parameter grid. The OpenAI documentation suggests changing only one of the two parameters.
- **Presence_penalty** must be set to 0 (default value) to reduce the "creativity" of GPT-4.
- **Frequence_penalty** must be in [0.0, 0.1, 0.2, 0.3, 0.4, 0.5] to reduce the model's proclivity to make repeated predictions.

[3]OpenAI documentation. https://platform.openai.com/docs/api-reference

- **Temperature** must be in [0.0, 0.1, 0.2, 0.3, 0.4] as far as it concerns reducing the creativity of GPT-4 and allowing it to be more deterministic.

Finally, we chose the number of folds for the cross-validation splitting strategy, namely four folds for the GPT-4 model (i.e., V1) refined in splitting sentences and for the model refined in COSMIC analysis (i.e, V2). To validate the results, we used the F1-score of the Rouge metric [89], [90], specifically 1-Rouge (see Appendix A2 for details).

The GridSearch process evaluated 25 combinations of temperature and frequency penalty values (5 frequency penalty values times five temperature values). For each combination, we computed the average F1-score across the four folds. The results showed that lower temperature values consistently produced better results, with temperature=0 yielding the highest F1-scores. This result aligns with our goal of making GPT-4's responses as deterministic as possible for COSMIC measurement. Similarly, a frequency penalty of 0 proved optimal, suggesting that penalizing word repetition does not improve performance in our context, likely because COSMIC measurement often requires consistent terminology.

The best settings for V1 and V2 identified by the GridSearchCV algorithm were a temperature of 0, presence penalty of 0, frequency penalty of 0, best of 1, and top p of 1. The Python sources to replicate the training and optimization pipeline are publicly available in the CosMet online repository [85].

3.5 The CosMet Tool

We developed a tool (see our online repository for sources [85]) implementing the CosMet approach, which interacts with the GPT-4 API through HTTP requests using the provided Python bindings [4]. It parses the GPT-4 responses and displays the output predictions on a web page. We used the Streamlit framework [5] – a Python web framework - to develop CosMet as a web application. Figures 13 and 14 show the tool's input and output.

In particular, CosMet accepts as input the textual representation of a use case and returns information about identified Triggering event, Functional user (FU), Data group (DG), Object of interest (OOI), and Data movement type (DM) and corresponding CFP for each sub-process.

3.6 CosMet Setup

Setting up CosMet for a new organization is straightforward and involves the following steps:

1) Reviewing organizational guidelines for writing use cases.
2) Identifying a representative set of use cases.
3) Fine-tuning the LLM using the annotated few-shot examples.
4) Validating the LLM's performance and making iterative improvements as needed.

The following effort estimates are based on the experience of the first author as a practitioner, who is certified

[4]OpenAI documentation. https://platform.openai.com/docs/api-reference

[5]Streamlit framework. https://streamlit.io/

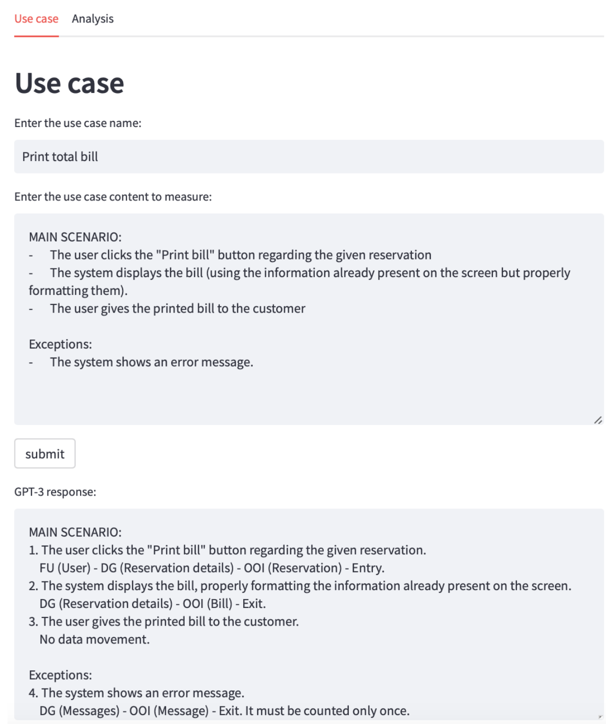

Fig. 13: CosMet user interface: use case

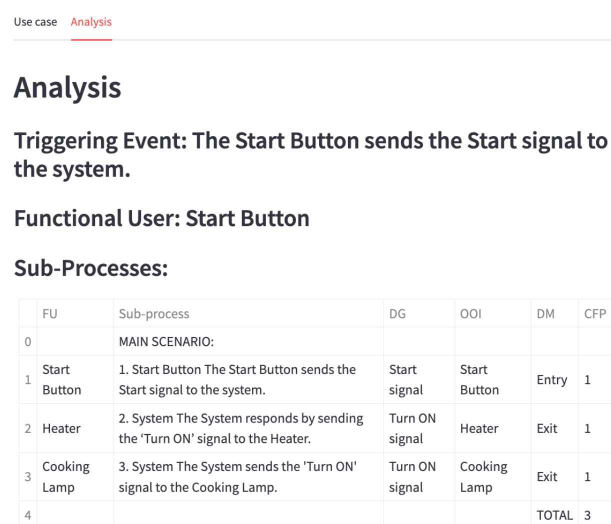

Fig. 14: CosMet user interface: COSMIC measurement analysis

in UML 2, COSMIC, and IFPUG, with over 30 years of experience in the IT field and more than 18 years specifically working with FSM across various projects in both public and private sectors. These estimates reflect real-world implementation scenarios rather than purely research-based settings.

The first step, typically 2 to 4 work hours, involves understanding specific terminologies, sentence structures, and formatting conventions unique to the organization. Indeed, it is essential to mention that companies typically establish guidelines for writing use cases [91]. These guidelines can significantly aid in selecting appropriate few-shot examples for COSMIC analysis. The next step is to identify a representative set of use cases that cover a broad range of functionalities and complexities. This phase usually takes about 4 to 8 work hours. Our experience with diverse domains has shown that this analysis ensures that the selected use cases are diverse and representative. The annotated few-shot examples are then used to fine-tune CosMet. This process, including iterative testing and validation, typically takes 8 to 12 work hours. The last step is validating the CosMet's performance using real-world use cases and making necessary adjustments based on feedback from professional measurers. Initially, this can take an additional 4 to 6 work hours, with periodic reviews as needed. The total time investment for the initial setup of CosMet typically ranges from 18 to 30 work hours, depending on the complexity of the use cases. While this setup phase requires an initial investment of time, it is a one-time effort that pays off in the long run. Once the setup is complete, the same few-shot examples can be reused across multiple projects, ensuring consistency and efficiency.

4 Empirical Assessment of CosMet

This section describes the research questions, dataset creation, and methods for evaluating the proposed approach.

4.1 Research Goals and Questions

The goal of the empirical assessment was to analyze the effectiveness of our approach (i) to measure use cases and (ii) to map the FURs in the form required by the COSMIC General Software Model, together with (iii) assess the amount of time required for COSMIC measurement.

The purpose was to provide empirical evidence that can highlight the benefits and limitations of the proposed approach to make practitioners aware of the strengths and weaknesses they would obtain through the use of CosMet. More specifically, our empirical assessment was driven by three main research questions, namely:

RQ1 How effective is CosMet in measuring use cases?
This research question focuses on CosMet's ability to identify Data Movements (Entries, Exits, Reads, and Writes), which represent the final output of COSMIC measurement.

RQ2 How effective is CosMet in mapping FURs to the COSMIC Generic Software Model?
This question evaluates CosMet's ability to correctly identify the key elements required by the COSMIC Generic Software Model: Functional Users, Triggering Events, Data Groups, and Objects of Interest. This detailed evaluation is crucial for ensuring transparency in our approach, allowing measurers to validate and, if necessary, intervene in the measurement process.

RQ3 How efficient is CosMet in reducing measurement time?
This question assesses how CosMet transforms the measurement process and its impact on the time required to perform COSMIC measurement compared with industry benchmarks.

To address the goal of our study, we followed the prompt engineering guidelines provided by OpenAI [83] and Ekin [84] and conducted mixed-method research, hence combining quantitative performance assessment with qualitative insights from practitioners. As for reporting, we followed the ACM/SIGSOFT Empirical Standards, particularly the "General Standard" and "Mixed Methods" guidelines.

4.2 Dataset Creation and Ground Truth

The datasets we utilized for our empirical assessment consist of 7 software systems from 4 different domains (Management Information System, Microservices/IoT, Real-Time, and Machine Learning), totaling 123 use cases and 119 functional processes:

- Albergate: A Management Information System provided by Kiranet (an SME Italian company), comprising 66 use cases and 62 Functional processes.
- FIDCPM, FID-MTC, and FID-TCT: Microservices of a Telemedicine application developed by Kiranet, providing 37 use cases and 37 Functional processes.
- Rise cooker and Automatic Line Switching: Case studies from the COSMIC Website [92], [93] in the Real-Time domain, comprising 10 use cases and 10 Functional processes.
- U-CURE: An ML application developed by Innovaway (a large Italian company), comprising 10 use cases and 10 Functional processes.

We engaged two professional COSMIC measurers certified in FSM methods (IFPUG and COSMIC) to create the ground truth. The first expert has over 25 years of experience in software engineering and more than 15 years in software effort/size estimation. The second is a manager of an IT company and a member of GUFPI - ISMA[6], with over 10 years of experience in software effort/size estimation.

The ground truth creation process followed three steps:

1) Independent measurement: Each expert independently analyzed and measured the use case specifications.
2) Comparison: The experts compared their measurements to identify discrepancies.
3) Fine-tuning: The experts conducted a fine-tuning session to resolve disagreements and produce shared COSMIC measurement reports.

We calculated the inter-rater agreement between the experts' independent measurements, specifically Cohen's Kappa score [94], before the fine-tuning session.

The experts agreed on 573 of 593 total movements for Data Movement identification, with only 20 disagreements (one in FID-CPM, 16 in FID-MTC, one in FID-TCT, and two in U-CURE). Cohen's Kappa showed almost perfect agreement (k = 0.976) in classifying movements as Entry, Exit, Read, or Write. The agreement levels for other COSMIC elements were also high: Triggering Events and Functional Users showed perfect agreement (k = 1.000). In contrast, Data Groups (k = 0.838) and Objects of Interest (k = 0.866) showed slightly lower agreement.

The lower agreement for Data Groups and Objects of Interest reflects the variations in terminology used by the experts (e.g., "Customer Data" vs. "Customer Information"), which were standardized during the fine-tuning session to ensure consistency. The resulting ground truth includes a

[6]https://www.gufpi-isma.org

comprehensive identification of data movements for each functional process, a complete mapping to the COSMIC Generic Software Model, encompassing functional users, triggering events, data groups, and objects of interest, as well as detailed measurement reports for each software system. All datasets and measurement reports are publicly available in our online repository [85].

4.3 Research method for RQ_1

To address RQ_1, we compared the COSMIC Data movements produced by COMET against the ground truth created by the expert measurers (described in Section 4.2).

For each software project, we evaluated use cases (in their original form as documented in the use case documents) at the functional process level, where each use case (including any included or extended use cases) was analyzed as a single unit. We then computed metrics to establish CosMet effectiveness. In particular, we used the BLEU [95] and Rouge (specifically 1-Rouge) [89], [90] metrics, explained in detail in Appendix A2. It is important to note that our evaluation task is not a simple classification problem but a comparison of ordered sequences that represent the flow of Data Movements within functional processes. For example, consider a functional process with four subprocesses where we identify the following sequence: [Entry, Read, Write, Exit]. A different sequence containing the same movements but in a different order, such as [Read, Entry, Exit, Write], would be completely wrong as it misrepresents the actual flow of the functional process. For BLEU, we used weights=(1, 0, 0, 0), which focuses on matching individual Data Movement labels at each position in the sequence, giving 100% weight to unigram matching. This configuration means that we evaluate each subprocess classification independently, comparing CosMet's output with the ground truth one position at a time. The Rouge-1 metric complements this evaluation by providing precision, recall, and F1-score for each type of Data Movement across the sequence. We performed 10 iterations of the evaluation process on the dataset to address the non-deterministic nature of LLMs. We executed CosMet on all use cases for each iteration and computed BLEU and Rouge-1 metrics. This repeated evaluation allows us to assess the stability and reliability of CosMet's Data Movement identification across multiple runs. We report the average scores and their standard deviations to provide a comprehensive view of CosMet's performance and any potential variations in its output. We used two Python libraries to evaluate BLUE and 1-Rouge metrics: nltk [7] and Rouge [8].

Additionally, to comprehensively assess the effectiveness of our approach, we systematically searched Scopus to identify all the articles proposing COSMIC automation tools and potential approaches to compare with CosMet. Among these, only the article by Ochodek et al. [96] used textual use cases, proposing a deep learning model for end-to-end approximation of COSMIC functional size. As such, this represented the baseline considered in our study. To compare CosMet with DEEP-COSMIC-UC, we proceeded as follows. Ochodek et al. initially assessed the accuracy of DEEP-COSMIC-UC by calculating absolute error (MAE) and median absolute error (MdAE). The values obtained in their study were 3.479 for MAE and 2.204 for MdAE. To ensure a fair comparison, we applied both CosMet and DEEP-COSMIC-UC to our dataset, computing MAE and MdAE for each of CosMet's 10 iterations against the ground truth, and for DEEP-COSMIC-UC's single output.

[7]Natural Language Toolkit for Python. https://www.nltk.org/
[8]Project Rouge. https://pypi.org/project/rouge/

4.4 Research method for RQ_2

To address RQ_2, we designed an evaluation approach consisting of two complementary phases: a broad quantitative evaluation and a qualitative assessment with practitioners.

4.4.1 Quantitative Evaluation

For the quantitative evaluation, we analyzed CosMet's performance in identifying Functional users, Triggering events, Data groups, and Objects of interest across our datasets (described in Section 4.2), which includes Albergate, FIDCPM, FID-MTC, Rice cooker, Automatic Line Switching, U-CURE, and FID-TCT use case documents. Note that while DEEP-COSMIC-UC serves as our baseline for Data Movement identification in RQ1, it was not included in this comparison as it does not provide capabilities for mapping FURs to the COSMIC Generic Software Model elements.

The evaluation employs three complementary metrics: BLEU [95], Rouge [89], [90], and BERTScore [97] (see Appendix A2 for details). For each COSMIC Generic Software Model element (Functional users, Triggering events, Data groups, and Objects of interest), we compared the sequence of elements identified by CosMet against the ground truth at the subprocess level, maintaining their sequential order (e.g., if for a subprocess CosMet identified the Data groups [”Room,” ”Customer,” ”Reservation”] and the ground truth contained [”Room,” ”Customer details,” ”Reservation”], we compared each Data group at its specific position). BLEU and Rouge capture the quality, precision, and recall of CosMet's output compared to the reference text from the ground truth (see Section 4.2), considering the sequential nature of the analysis where each subprocess contains interrelated COSMIC elements. BERTScore addresses semantic similarities, which is crucial when comparing CosMet's output with the ground truth, as semantically equivalent descriptions might be expressed using different words or phrasings (e.g., CosMet might identify a Functional user as ”system operator” while the ground truth refers to it as ”system administrator”). As in RQ_1, we performed 10 iterations to account for the non-deterministic nature of LLMs, reporting both average scores and standard deviations for all metrics.

4.4.2 Qualitative Evaluation

Since metrics cannot fully capture the correctness of COSMIC element identification, we performed a further analysis involving five practitioners with two years of experience with COSMIC measurement from an Italian software company and the two certified COSMIC measurers who created the ground truth to answer RQ_1.

In particular, we employed the use case document FID-TCT (described in Section 4.2) as a representative case study for this phase, as it encompasses typical characteristics found across our dataset while being of manageable size for detailed manual analysis. The practitioners used CosMet to analyze FID-TCT's use cases and evaluated the appropriateness of the generated Triggering events, Functional users, Data groups, and Objects of interest. To this end, the practitioners filled in questionnaire no. 1 shown in TABLE 2, where each question admits answers within a scale proposed by Likert, ranging from 1 to 5, where 1 means a meager value, and 5 indicates a very high value.

TABLE 2: Questionnaire no. 1 provided to the measurers.

No.	Questions
1	How appropriate are the CosMet identified Data groups? [1. Highly inappropriate, 2. Inappropriate, 3. Neutral, 4. Appropriate, 5. Highly appropriate]
2	How appropriate are the CosMet identified Objects of interest? [1. Highly inappropriate, 2. Inappropriate, 3. Neutral, 4. Appropriate, 5. Highly appropriate]
3	How appropriate are the CosMet-identified Functional users? [1. Highly inappropriate, 2. Inappropriate, 3. Neutral, 4. Appropriate, 5. Highly appropriate]

4.5 Research method for RQ_3

RQ_3 aims to assess how much time is required for COSMIC measurement and how much CosMet can reduce this time in a real-world scenario. This time includes the work the measurers need to evaluate, correct, and validate the CosMet measurement analysis, helping fine-tune the measurement and addressing any ambiguities or discrepancies in the requirements expressed by use cases. To establish a baseline for comparison, we referred to the industry benchmark reported by Ungan et al. [51], which indicates that a COSMIC-certified measurer typically measures 125-500 CFPs daily. In contrast, an uncertified measurer may measure less than half of this quantity. To assess this, we asked the five practitioners and the two certified professional measurers involved in the analysis to answer RQ_2, and evaluate how CosMet can help reduce measurement time. The participants recorded the total time to perform the CosMet analysis on the FID-TCT use case specifications (described in Section 4.2), fine-tuned it for each Functional process, and filled in questionnaire no. 2 shown in TABLE 3. We then collected and compared the participants' responses in order to understand the opinions of practitioners about how CosMet can reduce the time required for COSMIC measurement.

TABLE 3: Questionnaire no. 2 provided to the measurers.

No.	Questions
4	To what extent does CosMet reduce the measurement time compared to manual measurement? [1. 0%, 2. < 20%, 3. < 40%, 4. < 60%, 5. <80%]

In addition, to assess the CosMet's performance comprehensively, we calculated the measurement time of DEEP-COSMIC-UC using the same hardware configuration employed for CosMet and compared the results obtained.

5 Experimental Results

We present the results for each research question.

5.1 RQ_1 - How effective is CosMet in measuring use cases?

The average 1-Rouge and BLEU metrics across all the use case documents from the software projects in the four domains indicate high accuracy and consistency in the CosMet's analysis (see Table 4). Over 10 iterations, Rouge-1 showed 99.0% (SD = 2.7%) for precision, recall, F1-score, while BLEU reached 99.0% (SD = 3.2%). This means that, on average, CosMet correctly identified 587 out of 593 Data movements across all use cases, with minimal variation between iterations, demonstrating high reliability for automated COSMIC measurement. For the Albergate use cases, CosMet consistently identified 357 out of 359 Data movements across all 10 iterations, achieving 99.8% in Precision, Recall, F1-score (SD = 1.5%), and BLEU (SD = 1%).

For the use case documents of FIDCPM, FID-TCT, and FID-MTC, CosMet correctly identified an average 131.9 out of 134 Data movements, reaching 98.4% for precision, recall, and F1-score (SD=4.1%), and 99.0% for BLEU (SD=3.1%).

For the Rice cooker and Automatic Line Switching use case documents, CosMet correctly identified all 47 Data movements across all iterations, reaching 100% in all metrics (SD=0.00%). This result indicates that when comparing sequences of Data Movement types (i.e., Entry, Exit, Read, Write), both BLEU and 1-Rouge metrics indicate that CosMet identified the same sequence as the human measurers, without no differences in the type or order of the movements.

For the U-CURE use case document, CosMet correctly identified an average of 52 out of 53 Data movements out, reaching 98.3% in precision, recall, and F1-score (SD=5.3%), and 96.3% for BLEU (SD=11.9%).

The error analysis revealed several recurring patterns across domains. The first error type concerns distinguishing Data movements from control commands, as seen in Albergate's FP26. CosMet erroneously identified a control command as an Entry Data movement in the subprocess "The user clicks the 'View order details' button." According to the COSMIC manual, this represents a control command that enables user interaction without moving data about an Object of Interest. The error likely occurred due to the presence of action verbs ("clicks") and data-related terms ("details"), which resemble Data movement patterns.

The second error type concerns implicit Data groups requiring domain knolwledge. In Albergate's FP29 and U-CURE's FP10, CosMet failed to identify Entry and Exit Data movements related to implicit data (Customer data and SVC hyperparameters, respectively). For example, in U-CURE's subprocess "The system requests the ML Engine to classify the patient," the Exit Data movement for SVC parameters was missed. This suggests that CosMet struggles to infer implicit data movements that experienced measurers would recognize from context.

A third, albeit rare, error type involves sentence splitting issues. In FID-MTC's FP6, one iteration failed to split the compound sentence "The system searches and displays all the televisit requests for the given doctor received from his (her) patients" into separate search and display subprocesses. However, CosMet demonstrated robustness by still correctly labeling it as "Read/Exit," accurately identifying both Data movements despite the splitting issue.

TABLE 4: Quantitave Evaluation Results.

			MIS	IoT / Microservice			Real time		AI	
COSMIC Concept	Metric		Albergate	K01719	K01720	K01726	ALS	Rise Cooker	U-CURE	AVG
TE	1-Rouge	F1-Score	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
		Std.Dev.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	BLEU	Val.	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
		Std.Dev.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	BertScore	F1-Score	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
		Std.Dev.	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
FU	1-Rouge	F1-Score	0.998	0.987	1.000	0.991	1.000	1.000	1.000	0.996
		Std.Dev.	0.015	0.034	0.000	0.035	0.000	0.000	0.000	0.012
	BLEU	Val.	0.996	0.988	1.000	0,987	1,000	1.000	1.000	0.996
		Std.Dev.	0.030	0.032	0.000	0.050	0.000	0.000	0.000	0.016
	BertScore	F1-Score	0.999	0.995	1.000	0,996	1.000	1.000	1.000	0.999
		Std.Dev.	0.009	0.012	0.000	0.017	0.000	0.000	0.000	0.005
DG	1-Rouge	F1-Score	0.917	0.913	0.958	0.948	0,986	1.000	0.926	0.950
		Std.Dev.	0.099	0.120	0.084	0.106	0.031	0.000	0.127	0.081
	BLEU	Val.	0.847	0.905	0.945	0.930	0.979	1.000	0.895	0.929
		Std.Dev.	0.170	0.125	0.107	0.129	0.046	0.000	0.194	0.110
	BertScore	F1-Score	0.967	0.977	0.987	0.982	0.994	1.000	0.983	0.984
		Std.Dev.	0.031	0.029	0.025	0.031	0.013	0.000	0.029	0.023
OOI	1-Rouge	F1-Score	0.890	0.837	0.987	0.793	0.949	0.689	0.801	0.849
		Std.Dev.	0.163	0.299	0.052	0.236	0.093	0.153	0.316	0.188
	BLEU	Val.	0.818	0.817	0.987	0.735	0.914	0.624	0.771	0.809
		Std.Dev.	0.235	0.318	0.030	0.259	0.155	0.290	0.328	0.231
	BertScore	F1-Score	0.966	0.972	0.996	0.950	0.983	0.956	0.958	0.969
		Std.Dev.	0.040	0.049	0.010	0.047	0.030	0.050	0.051	0.040
DM	1-Rouge	Precision	0.998	0.961	1.000	0.992	1.000	1.000	0.983	0.990
		Std.Dev.	0.015	0.091	0.000	0.032	0.000	0.000	0.053	0.027
		Recall	0.998	0.961	1.000	0.992	1.000	1.000	0.983	0.990
		Std.Dev.	0.015	0.091	0.000	0.032	0.000	0.000	0.053	0.027
		F1-Score	0.998	0.961	1.000	0.992	1.000	1.000	0.983	0.990
		Std.Dev.	0.015	0.091	0.000	0.032	0.000	0.000	0.053	0.027
	BLEU	Val.	0.998	0.982	1.000	0.989	1.000	1.000	0.963	0.990
		Std.Dev.	0.010	0.052	0.000	0.039	0.000	0.000	0.119	0.032

TE=Triggering Event FU=Functiona User, DG=Data Group, OOI=Object Of Interest, DM=Data Movement

The fourth pattern emerges in user interaction scenarios, particularly in FIDCPM and FID-MTC. In FIDCPM's FP4, CosMet consistently swapped Data movements in ”The Sensor Device sends the raw data to the Mobile” (identified as Exit instead of Entry) and ”The Mobile App requests the System to save the measurement data” (identified as Entry instead of Exit). Similarly, in FIDCPM's FP5 and FID-MTC's FP13, FP14, and FP21, the model struggled with entry movements in button-click scenarios such as ”The user selects the 'List all vitals' button.” These errors suggest that CosMet may be overly influenced by sentence structure and specific verbs rather than the underlying data flow logic.

These patterns indicate that while CosMet maintains high overall effectiveness, it faces challenges with complex semantic distinctions, implicit domain knowledge, and specific syntactic patterns. These limitations primarily stem from the restricted number of examples used during the refinement phase, constrained by GPT-4's context size limits. Additional few-shot examples focusing on these specific patterns could improve performance in these areas.

Despite minor discrepancies in a few Functional processes, the results demonstrate that CosMet is highly effective in measuring use cases within the considered domains. Tables in Appendix B.1 report detailed results in terms of CFPs obtained with CosMet and by manual measurement for each considered system and each identified Functional process.

Regarding the comparison with DEEP-COSMIC-UC [96], the MAE and MdAE values confirmed the results above. In particular, across all iterations, CosMet's MAE scores ranged from 0.026 to 0.053 (average = 0.038), compared to 2.8421 for DEEP-COSMIC-UC, while MdAE values were 0 for CosMet and 2 for DEEP-COSMIC-UC. These results indicate that CosMet's predictions almost perfectly match the expert measurers' ground truth, with an average deviation of only 0.038 Data movements per functional process, while DEEP-COSMIC-UC shows a larger average deviation of about 3 Data movements. The above discussed results allow us to answer RQ_1 as follows:

> *CosMet is a good starting point for manually fine-tuning the measurement because it displays the interpreted data movements of each FUR; the user has total transparency into the makeup of the count and may properly change the requirements (i.e., ambiguity, lack of clarity).*

5.2 RQ_2— How effective is CosMet in mapping FURs to the COSMIC Generic Software Model?

In the following, we first compare the CosMet-generated text (regarding Functional users, Triggering events, Data groups, and Objects of interest) against the ground truth, using the 1-Rouge, BLEU, and BERTscore metrics. Then, we analyze the answers to questionnaire no. 1 filled in by the five practitioners and the two certified COSMIC measures involved in the analysis of RQ_1, considering the results of the evaluation they performed on the FID-TCT use case document.

5.2.1 CosMet analysis vs manual measurement

The average 1-Rouge and BLEU metrics values for all the use case documents of the software projects from the four domains indicate a high level of performance (see Tables 4 and 5 and Appendix C1 for additional details). Specifically, the BLEU scores range from 80.1% to 100% (SD=9.7%), the F1-Score values for the 1-Rouge metric vary between 84.5% and 100% (SD=7.4%), and the F1-Score values for the BERTscore metric range from 96.6% to 100% (SD=1.8%) across all iterations.

For the Albergate system (comprising 359 Data groups and 24 Objects of interest), CosMet perfectly identified Triggering events, reaching 100% across all metrics (SD=0.0%). For Functional Users, the system demonstrated exceptional performance with Rouge metrics showing precision, recall, and F1-score values of 99.8% (SD=1.5%), BLEU score reaching 99.6% (SD=2.9%), and BERTscore achieving 99.9% across all measures (SD=0.9%). Regarding Data Groups, CosMet maintained strong performance, with Rouge metrics showing precision at 93.9% (SD=8.4%), recall at 90.4% (SD=11.1%), and F1-score at 91.7% (SD=9.9%). While the BLEU score was lower at 84.7% (SD=17.0%), BERTscore demonstrated robust results with all measures above 96% and lower variability (SD≈3%). For Objects of Interest, Rouge metrics achieved precision at 91.7% (SD=16.1%), recall at 87.9% (SD=16.9%), and F1-score at 89.0% (SD=16.3%). While the BLEU score decreased to 81.8% (SD=23.5%), BERTscore maintained excellent results above 96% (SD≈4%).

As for the use case documents of FIDCPM, FID-TCT, and FID-MTC (characterized by 134 Data groups and 18 Objects of interest), CosMet correctly identified the Triggering events and Functional users for all the use cases, reaching 100% in all metrics. Regarding Data Groups, performance was consistent across metrics, with Rouge achieving precision, recall, and F1-score all around 94% (SD≈10%). BLEU score reached 92.7% (SD=12.1%), while BERTscore showed the best results with all measures above 98% and lower variability (SD≈3%). For Objects of Interest, performance showed more variability, with Rouge metrics achieving precision at 86.6% (SD=19.8%), recall at 87.9% (SD=19.8%), and F1-score at 86.9% (SD=19.6%). While BLEU score was lower at 84.6% (SD=20.2%), BERTscore maintained strong results above 97% (SD≈4%).

Similarly to previously considered domains, CosMet correctly identified the Triggering events and Functional users for all the use cases of Rice cooker and Automatic Line Switching (characterized by 47 Data groups and 8 Objects of interest), achieving 100% in all metrics. Regarding Data Groups, performance was exceptional across all metrics. Rouge showed outstanding results with precision at 99.2% (SD=1.3%), recall at 98.9% (SD=1.8%), and F1-score at 99.0% (SD=1.5%). BLEU score achieved 98.5% (SD=2.3%), while BERTscore demonstrated near-perfect performance with all measures at 99.6% and minimal variability (SD=0.6%). For Objects of Interest, performance remained strong but showed more fluctuation. Rouge metrics achieved precision at 87.7% (SD=10.7%), recall at 86.7% (SD=13.2%), and F1-score at 87.1% (SD=12.3%). While the BLEU score was lower at 82.7% with higher variability (SD=22.3%), BERTscore maintained excellent results with consistent 97.5% across all measures (SD≈4%).

Concerning the fourth domain, CosMet correctly identified the Triggering events for all use cases of U-CURE (comprising 52 Data groups and 9 Objects of interest), reaching 100% in all metrics. It correctly determined the Functional users in all Functional processes, reaching 100% in all metrics. Regarding Data Groups, performance remained strong with Rouge metrics showing precision at 94.6% (SD=9.2%), recall at 91.7% (SD=15.3%), and F1-score at 92.6% (SD=12.7%). While the BLEU score reached 89.5% (SD=19.4%), BERTscore maintained excellent results with all measures above 98% and lower variability (SD≈3%). For Objects of Interest, Rouge metrics reached 80.1% across precision, recall, and F1-score (SD=31.6%). The BLEU score achieved 77.1% (SD=32.8%), while BERTscore maintained higher performance with precision at 95.4% (SD=5.6%), recall at 96.2% (SD=4.7%), and F1-score at 95.8% (SD=5.1%).

These results indicate that while CosMet provides reliable identification of COSMIC elements and a valuable starting point that significantly reduces the manual effort required for COSMIC measurement, manual verification might be beneficial, particularly for Objects of Interest where average F1-scores range from 80.9% (BLEU) and 84.9% (Rouge) to 96.9% (BERTscore). A deeper analysis revealed several identification challenges across all domains that could explain these variations. Initially, during the very first iteration, CosMet struggled with Functional User identification in multiple cases: nine Functional processes in Albergate (FP 39, 43, 46, 57-62) and twenty-one in FIDCPM and FID-MTC (FP 1, 2, 5-9 in FIDCPM and FP 1, 2, 4-6, 8, 10-15, 18, and 23 in FID-MTC). The problem was primarily due to the generic references to users as "User" or "Actor" in use case descriptions, as shown in the following example:

The user clicks the "View order details" button for the order of his (or her) interest.

When these use cases were modified to specify the Functional User names explicitly, CosMet correctly identified all Functional Users, achieving perfect scores across all metrics. Additional issues in Functional user identification emerged in two specific cases. In FP26 of Albergate, as reported in Section 5.1, CosMet erroneously identified a control command as an Entry Data movement, reporting an additional Functional User, while in FP29, it failed to identify an Entry Data movement, resulting in a missing Functional User. Regarding Data Groups and Objects of Interest identification, we observed both splitting and recognition challenges. In one iteration for FID-MTC's FP6 (as reported in Section 5.1), the splitter component failed to properly separate a compound sentence, resulting in missed identification of a related Object of Interest and its associated Data Group. Additionally, the Rouge metric values were particularly low (<60%) in a few cases: six Functional processes in the MIS domain, four in the IoT/Microservice domain, one in the Real-time domain, and one in the AI domain. Interestingly, the BERTscore metric performed significantly better for these same processes, with F1 scores ranging from 85% to 93%. This discrepancy can be attributed to ambiguities in the use case requirements. For instance, we found cases where synonyms were used for the same Object

TABLE 5: Evaluation Results by COSMIC Concept.

COSMIC Concept	Rouge Precision (Std.Dev.)	Rouge Recall (Std.Dev.)	Rouge F1-score (Std.Dev.)	BLEU (Std.Dev.)	BERTscore Precision (Std.Dev.)	BERTscore Recall (Std.Dev.)	BERTscore F1-score (Std.Dev.)
Triggering event	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)	1.000 0.000
Functional User	0.998 (0.009)	0.998 (0.009)	0.998 (0.009)	0.997 (0.014)	0.999 (0.005)	0.999 (0.005)	0.999 (0.005)
Data Group	0.955 (0.073)	0.939 (0.096)	0.944 (0.086)	0.915 (0.127)	0.984 (0.022)	0.981 (0.026)	0.982 (0.024)
Object of Interest	0.853 (0.196)	0.843 (0.204)	0.845 (0.200)	0.801 (0.247)	0.967 (0.042)	0.966 (0.042)	0.966 (0.042)
MIN	**0.853**	**0.843**	**0.845**	**0.801**	**0.967**	**0.966**	**0.966**
MAX	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
STD.DEV.	**0.070**	**0.077**	**0.074**	**0.097**	**0.017**	**0.018**	**0.018**

of Interest (e.g., ”Feature” instead of ”Categorical feature” in U-CURE). Due to GPT-4’s context size limitations, we could not include additional few-shot examples to improve synonym identification.

5.2.2 Appropriacy evaluation

The analysis of responses to questionnaire no. 1 indicated a strong agreement among the interviewees (see Table 6 and Apendix C2).

TABLE 6: Participants’s answers to the questionnaire no 1.

Participant	Question no. 1	Question no. 2	Question no. 3
1	4	3	4
2	5	4	4
3	4	4	4
4	4	4	5
5	5	4	5
6	5	4	5
7	5	4	5
MEAN	**4.6**	**3.9**	**4.6**
MEDIAN	**5**	**4**	**5**
STD. DEV.	**0.53**	**0.38**	**0.53**

In particular, it revealed that the mean scores for questions 1 and 3 are both 4.6, with a median of 5, suggesting that participants generally rated these aspects (i.e., appropriacy in identifying Data groups and Functional users) of CosMet highly. Question 2 (about appropriacy in identifying Objects of interest) has a slightly lower mean score of 3.9 and a median of 4, indicating a positive reception. The standard deviation for questions 1 and 3 is 0.53, while for question 2 it is 0.38, which shows a relatively small spread in the responses, further confirming the consistency of the positive feedback.

These findings corroborate the earlier sections, highlighting the effectiveness of CosMet in accurately identifying relevant components for the users, and lead us to answer RQ_2 as follow:

> *CosMet can successfully map the FURs to the form required by the COSMIC Generic Software Model because it correctly identifies Functional users, Triggering events, and Data Movements in use case specifications. Furthermore, it performs fine in recognizing Data groups and Objects of interest and generates excellent qualitative mapping.*

5.3 RQ_3— How efficient is CosMet in reducing measurement time?

The study conducted on the FID-TCT microservice application, which has a total functional size of 17 CFPs, revealed that the average durations for each Functional process, as reported by participants, varied. The aggregated data from the participants’ measurements indicated that the average time to complete the CosMet approach for a single Functional process ranged from 19.27 to 62.52 seconds. When considering the entire application, the total average duration for executing the CosMet approach was 198.21 seconds. Additionally, the time required for fine-tuning the analysis was recorded at 258.29 seconds. Therefore, the overall execution time for the CosMet approach, including fine-tuning, amounted to 456.49 seconds.

These timing results reflect a significant transformation of the measurement process. In traditional manual measurement, practitioners need to (1) read and understand each use case, (2) identify all COSMIC elements (Functional Users, Data Groups, Objects of Interest), (3) count and classify Data movements, (4) document the measurement, and (5) review their work. With CosMet, this process is streamlined to (1) running CosMet on the use cases, (2) reviewing and fine-tuning CosMet’s output, and (3) validating the results. To put these results in perspective, a COSMIC-certified measurer typically processes 125-500 CFPs daily [51], while uncertified measurers achieve less than half this rate. CosMet can reduce measurement time by 53.38% to 88.35%, enabling the analysis of approximately 1,072.53 CFPs within an eight-hour workday. This efficiency gain was confirmed by practitioners, with 71% indicating a 60% reduction in measurement time, and 14% reporting even greater time savings. Participant feedback through questionnaire no. 2 showed consensus on the 60% time reduction, with an average response score of 4. Detailed measurement durations and response distributions are available in Table 7 and Figure 15 (see Appendix D for additional details).

When compared to DEEP-COSMIC-UC [96], which averages 4.23 seconds per use case and 10.19 seconds for the entire FID-TCT model, CosMet is slower ($\approx$ 19 times). However, while DEEP-COSMIC-UC only estimates CFPs, CosMet provides comprehensive COSMIC analysis and measurement results, offering greater transparency and allowing practitioners to refine use-case documents and fine-tune the analysis.

TABLE 7: Measurement durations recorded by the participants for FID-TCT.

Functional Process No.	Average COMET duration (seconds)
1	62.52
2	51.4
3	23.59
4	41.41
5	19.27
(A) TOTAL COMET Time	**198.21**
(B) FINE-TUNING Time	**258.29**
(A+B) TOTAL Time	**456.49**
CFP per second	**0.04**
TOTAL Number of CFPs in 8 hours	**1,072.53**

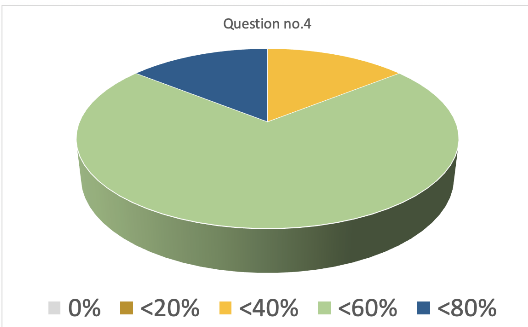

Fig. 15: Time reduction using CosMet according to the measurers.

All this leads us to answer RQ_3:

> *CosMet can significantly reduce the measurement time by executing the analysis in minutes. This time includes the work the measurer needs to evaluate, correct, and validate the CosMet measurement analysis.*

6 Discussion, Implications, and Limitations

In the following, we first discuss the practical implications of CosMet and outline directions for future work. Then, we analyze the threats to validity that may have influenced our findings and how we mitigated them.

6.1 Implications and Future Work

Our primary contribution is establishing a foundational framework for integrating LLMs into the COSMIC FSM process. While comparing COSMIC with other FSM methods like FPA could provide valuable insights, this work deliberately focused on COSMIC automation to avoid introducing complexities related to the inherent differences between FSM methods. Such comparative analysis remains an interesting direction for future research.

A key practical implication of our work is the transformation of the measurement workflow through CosMet. Traditional manual measurement requires practitioners to perform five distinct steps: (1) read and understand each use case, (2) identify all COSMIC elements, (3) count and classify Data movements, (4) document the measurement, and (5) review their work. CosMet streamlines this into three steps: (1) running the automated analysis, (2) reviewing and fine-tuning the output, and (3) validating the results.

This efficiency gain is substantial - our empirical assessment shows that CosMet enables the analysis of approximately 1,072 CFPs within an eight-hour workday, representing a time reduction of 53-88% compared to traditional manual measurement.

The practical benefits of CosMet extend beyond time savings. Organizations can leverage CosMet to standardize their measurement process, reducing the variability typically introduced by different measurers' interpretations. The tool's structured output provides transparency in the measurement process, allowing practitioners to trace how each measurement was derived and make informed adjustments where necessary. This aspect is particularly valuable for training new measurers, as they can learn from examining CosMet's detailed analysis and reasoning.

CosMet's ability to process natural language requirements also has significant practical implications for modern development practices. In Agile environments, where requirements frequently evolve and are often expressed as user stories, CosMet can provide rapid measurement feedback, enabling more accurate and timely effort estimation. The tool's quick processing time allows organizations to measure requirements as they are written, supporting more dynamic planning and estimation processes.

Several promising directions for future research emerge from our practical experience with CosMet. First, investigating the integration of CosMet with existing project management and estimation tools could enhance its practical utility in development workflows. Additionally, exploring how CosMet could be adapted to handle different requirement formats (beyond use cases) would increase its applicability across different development methodologies. Furthermore, studying how practitioners interact with and learn from CosMet's detailed analysis could improve organizational measurement training and knowledge transfer. Finally, investigating how CosMet could be combined with other estimation approaches in an ensemble strategy could lead to more robust prediction models. This ensemble approach could improve accuracy, adaptability, and robustness in software development effort estimation, enhancing prediction reliability and scalability in different project contexts.

6.2 Threats to validity

This section discusses possible limitations that could have biased our findings and how we tried to mitigate them.

6.2.1 Threats to construct validity

In our study, we identified potential construct validity threats. The first one concerns the metrics for assessing CosMet's performance. We combined BLEU, Rouge, and BERTScore metrics with the evaluation of certified COSMIC experts, ensuring technical and practical assessment. A second threat concerns mistakes and subjectivity in manual measurement, potentially impacting the ground truth used as a reference for evaluating CosMet. We involved two certified COSMIC professionals to mitigate this threat. They independently analyzed and measured the use cases and then performed a fine-tuning session to arrive at shared COSMIC measurement reports for each use case document.

6.2.2 *Threats to internal validity*

The main threats to internal validity relate to our experimental setup and controls. First, regarding CosMet's efficiency measurement, we mitigated potential bias by collecting timing data from all seven participants working on the same FID-TCT use cases, ensuring a controlled assessment. The selection of hyperparameters for GPT-4 could also affect our results. As detailed in Section 3.4, we controlled this threat by employing the GridSearch algorithm with Cross-Validation to identify optimal parameter settings. Preparing few-shot examples represents another potential threat to internal validity. We mitigated this by selecting examples from the COSMIC C-REG case study and engineering additional examples to cover batch processes and typical message exchanges with external components, as described in Section 3.2. The effectiveness of these examples was validated through the preliminary validation sets V1 and V2, ensuring their appropriateness for COSMIC measurement.

6.2.3 *Threats to external validity*

External validity concerns the generalizability of our findings to different contexts and situations. The primary threat relates to the representativeness of our dataset. While we included use cases from multiple domains, CosMet's performance might vary when applied to use cases from other domains or with different characteristics. This aspect was evident in our results, where domain-specific terminology impacted performance. For example, in the AI domain, where terms like "Feature" versus "Categorical feature" led to lower Rouge metric values (<60%) compared to other domains. Similarly, technical terms related to sensor data and measurements occasionally resulted in identification challenges in the IoT domain. We partially mitigated this threat by including diverse use cases from four domains and by providing a systematic approach for domain adaptation through a few-shot examples selection (as detailed in Section 3.6) . We also made our datasets publicly available in our online repository to enable further validation across different contexts. The non-deterministic behavior of LLMs challenges generalizability, as GPT-4's responses may vary between runs. We mitigated this threat by performing 10 iterations per experiment and reporting average and standard deviations (Sections 4.3, 4.4, 5.1, 5.2.1) while controlling GPT-4's behavior through hyperparameter tuning (Section 3.4). The comparison with DEEP-COSMIC-UC as the sole baseline may seem limited. However, as noted in Section 4.3, it was the only available tool for COSMIC measurement from textual use cases. We mitigated this issue by providing detailed metrics and sharing our experimental materials. Finally, domain-specific terminology may affect CosMet's performance across different domains. However, as detailed in Section 3.6, we mitigated this threat by selecting domain-representative few-shot examples during setup. We also provided a clear workflow to set up CosMet for a new organization, ensuring robust cross-domain adaptation.

6.2.4 *Threats to conclusion validity*

Conclusion validity concerns the reliability of our conclusions and the statistical relationship between treatment and outcome. A primary threat relates to GPT-4's token limit of 8,192 tokens, which could affect the reliability of our results when processing large or complex use cases. As described in Section 3.2, we mitigated this by carefully managing token allocation during refinement, leaving over 5,200 tokens to process a use case and produce the COSMIC analysis. The statistical reliability of our results might be affected by the sample size of use cases and the number of practitioners involved in the evaluation. Our dataset included 123 use cases across domains and seven practitioners. To provide transparency about the variability in our findings, we reported standard deviations for all our measurements across the 10 iterations performed for each experiment. The reliability of our results might be affected by both the practitioners' characteristics and their familiarity with CosMet. We involved seven measurers (five COSMIC-experienced professionals and two certified measurers) and controlled learning effects by using the simpler FID-TCT model for timing data collection. As shown in Section 5.3, time savings were consistent across all participants.

7 Related Work

In this section, we first examine previous research on measuring COSMIC using NLP. Then, we review the studies on automated COSMIC measurement. While our work focuses on automating COSMIC measurement, it is essential to note that other size measurement approaches exist, such as use case points, story points, object-oriented measures, requirements complexity, interface complexity, and SLOC. These different approaches highlight the diverse needs of software projects and represent potential areas for future research using similar methods.

7.1 COSMIC measurement using NLP

Simplifying and automating the application of the COSMIC method is crucial for both academia and industry, especially when measuring textual requirements. However, there is limited academic research on this topic.

Hussain et al. [50] present a workbench, based on a text miner, for automatically extracting all the necessary information from a software's requirement specifications document to calculate its COSMIC size measure. Hussain et al. [56] describe a supervised text mining strategy for estimating COSMIC functional size (based on [50]) from informally expressed textual requirements, with applications in Scrum and similar agile software development processes. The experiments' average F1-score value was 66.9%. Hussain et al. [55] use the frequency of language characteristics to infer the COSMIC size. However, the proposed method performs better when requirements are not represented as use cases or scenarios. The results show that it correctly classifies requirements with an F1-score value between 60.9% and 73.3%. Ochodek [53] uses NLP to parse linguistic elements from requirements and applies the C4.5 decision tree classifier [99] to categorize based on a historical database of requirements measurements. Requirements are

TABLE 8: Comparison with Related Work on COSMIC Measurement using NLP

Reference	Use Cases	Approach	Completeness	Limitations	Accuracy	Measurement time reduction
CosMet	✔	NLP with generative AI	Measure all scenarios within a use case	Layer and measurement scope cannot be identified	F1-score is 99.4%	60-80%
Hussain et al. [50]	✖	Text Mining	Only Data Groups and Data movements are identified	No experimental results	N/A	Not declared
Hussain et al. [56]	✖	Supervised text mining strategy for estimating COSMIC functional size (based on [6] and [7]).	Only CFPs are estimated	(1) Use historical data to estimate the COSMIC size; (2) Requirements are expressed as user stories.	66.9%.	Not declared
Hussain et al. [55]	✖	It uses the frequency of language characteristics to infer COSMIC size.	Only CFPs are estimated	(1) Use historical data to estimate the COSMIC size; (2) Do not perform well with use case scenarios.	F1-Score within 60.9% and 73.3%.	Not declared
Ochodek [53]	✔	It extracts syntactic linguistic information using NLP and classifies requirements using the C4.5 decision tree-based classifier.	Only CFPs are estimated	Sensitive to the language used in descriptions of requirements.	38.7%-78.5% of manual count.	Not declared
Ochodek [54]	✔	Two-step approach for predicting the functional size of applications based on use-case requirements.	Only CFPs are estimated	Use historical data to estimate the COSMIC size.	79%.	Not declared
Ecar et al. [98]	✖	Based on a User Story Grammar Validator and a Parser, and defines a dictionary to map it to the COSMIC method.	Only CFPs are estimated	(1) Only valid for User stories; (2) Use historical data to estimate the COSMIC size.	95.3%.	Not declared
Ungan et al. [51]	✖	Commercial tool for measuring textual requirements	Measure only the main function within a requirement	Syntactic and ontologic analysis of requirements; analyze only the main function of the requirements.	20%-30% of manual count.	75-80%
Wang et al. [52]	✖	Technique based on fixed language characteristics and syntactic rules.	Triggering events, Data movement types, and Data groups are not identified	Focused only on grammar and syntax rules and structured text	93%-99% of manual count.	90%

sorted into size classes defined by quantiles of the CFP distribution. Classification accuracy is influenced by the specification language, with the HKO predictor achieving 38.7% to 78.5% accuracy. Ochodek [54] proposes a two-step method for estimating the functional size of software from use case requirements. The process involves classifying use case names into thirteen categories and constructing prediction models using historical data to determine the number of CFPs. Ochodek's approach, compared with the COSMIC AUC and another technique [53], achieves a 79% accuracy rate. Ecar et al. [98] introduce AUTOCOSMIC, an automated tool for estimating user stories using a User Story Grammar Validator and Parser linked to the COSMIC method. Informal validation against a certified expert's estimates shows AUTOCOSMIC achieves a 95.3% accuracy rate, similar to an expert professional. SCOPEMASTER [51] is a COSMIC Functional Size Measurement tool that accepts textual requirements as input. It is based on the syntactic and ontologic analysis of the textual requirements. It analyzes only the primary function provided in the requirements. Its accuracy is within 20-30% of manual count equivalents, and it reduces the manual measurement time by 75%-80%. Wang et al. [52] offer an enhanced COSMIC software assessment technique based on fixed language characteristics, NLP concepts, and a local rule basis to eliminate human intervention and increase accuracy. To perform automated text processing, it focuses on grammar and syntax rules. The results show an accuracy within 93%-99% of manual count equivalents.

CosMet differs from the mentioned related work in several significant ways (see TABLE 8). Firstly, CosMet employs use case documents as input, while many related works focus on semi-structured textual requirements or user stories. Secondly, CosMet employs advanced NLP techniques, specifically LLMs, to automate the COSMIC measurement, allowing it to adapt to different contexts and styles of writing, potentially making it more flexible and accurate in different scenarios. The related work, on the other hand, often relies on predefined pattern rules or syntax-based classifiers, which may not be as adaptable to different contexts or styles of writing. Thirdly, CosMet delivers a comprehensive analysis of COSMIC concepts within functional requirements, offering greater transparency and detail than similar works. Finally, CosMet's performance has been validated across seven use case documents from diverse domains and by seven practitioners, showing high accuracy and notable productivity improvements. Such an extensive empirical evaluation and evidence of efficiency gains are often absent in related studies. Therefore, CosMet introduces a more sophisticated, automated, and detailed COSMIC measurement approach with proven effectiveness, customization potential, and substantial productivity benefits.

7.2 Automatic COSMIC measurement

Several researchers have suggested methods for automatically evaluating COSMIC measurements using various artifacts such as UML diagrams, conceptual models, structured requirements, source code, and more.

Jenner [46] provides a mapping between the COSMIC and Rational Unified Process concepts and designs a method to measure models automatically. Jenner [44] details an automated approach for determining functional size by utilizing Rational Rose software. On the same page, Azzouz and Abran [45] propose utilizing the Rational Unified Process and Rational Rose to automate the COSMIC functional size measurement. In [47], Diab et al. present μcROSE, a system that determines the functional size of real-time systems starting from statechart diagrams. The approach relies on programmed rules that map the several objects of interest onto the COSMIC ones. Lind et al. [48] introduce a technique for measuring embedded software using the COSMIC method. They propose a new UML profile containing all the information required to use COSMIC and estimate the functional size.

Soubra et al. [39], [40] design and implement an automated measuring procedure utilizing the most commonly used Simulink model in the automobile sector. Oriou et al. [41] describe how to use the approach and the tool by Soubra et al. within Renault. An automated approach for ensuring consistency between component and activity diagrams is designed by Sellami et al. [42], utilizing COSMIC. At first, the authors outline the methods used to establish the functional size of the diagrams. Then, rules are introduced to ensure the COSMIC FSM remains consistent. Lastly, a tool is offered to calculate the functional size and ensure the consistency of the diagrams.

Akca and Tarhan [43] develop a library to tag data movements in functional processes. The measuring approach is semi-automatic, requiring source code modification to use the library. According to the authors, using the library in a student registration system resulted in 92% accurate outcomes in automatically and manually measured functional size. Ungan et al. [49] establish a link between the abstraction levels of the measured artifacts and functional size. They provide a software size measuring approach based on software design model sizes for the solution domain. The suggested method, in particular, makes use of sequence diagrams. They also suggest an automated measurement tool. A novel tool, named PL FSM, has been introduced by Eren et al. [37] for facilitating functional software size measurement in a structured component-based software product line environment (CBPL) that utilizes the interface-based design (IbD) method. PL FSM establishes a mapping between UML and COSMIC elements and offers automated information extraction capabilities from UML diagrams. Implementing PL FSM streamlines the software development process in a CBPL framework. Gonultas and Tarhan [36] introduce a method for measuring the COSMIC functional size that automatically installs measurement library code into an application. This technique estimates the size while the user scenarios are executed at run-time.

Ceke and Milasinovic [35] describe a technique to approximate the web applications' functional size. This approach relies on a methodological mapping between the UML-based Web Engineering (UWE) models of web applications and the COSMIC functional size. Automated measurement outcomes are comparable. Karim et al. [32] suggest a tool for analyzing the XML structure of sequence diagrams to automate the measuring procedure. Tarhan et al. [38] delve into the automation of FSM through software code and introduces the development of a new tool called COSMIC SOLVER. This tool is specifically designed to measure the performance of Java Business Applications (JBAs). COSMIC SOLVER automates the entire process, from extracting textual representations of UML sequence diagrams from the functional execution traces of a JBA to tagging these representations with AspectJ to determine the COSMIC functional size. Furthermore, the tool calculates the functional size of user scenarios that run in the JBA, using the tags information following the COSMIC rules. Chamkha et al. [33] design a JAVACFP plugin tool for calculating the COSMIC functional size of Java source code. With JAVACFP, it is possible to verify the adequacy of implemented functions concerning the specified requirements, detect any inconsistencies, and generate updates on the progress of newly implemented functions. This plugin was validated using the *C-Reg* case study [86]. Zaw et al. [31] provide a model and an automated method for measuring it based on three alternative diagram notations, including UML, SysML, Petri net, and generic mapping. De Vito et al. [30] present a tool, J-UML COSMIC, for calculating the COSMIC functional size using UML software artifacts. The tool can deal with various UML artifacts based on the observation that different development processes can use different UML models. De Vito and Ferrucci [100] introduce Quick/Early, a streamlined process for measuring use case documents to balance accuracy with time constraints. They also suggest a template to easily identify functional components in use cases. Soubra et al. [29] present an approach for creating a COSMIC 'universal' tool for automatically measuring software developed in various programming languages. This study proposes a prototype tool based on COSMIC and MIPS with a small-scale validation as a proof of concept.

Furthermore, many practitioners and researchers use alternative and supplemental UML or SysML model types to be translated into software cost model parameters. For instance, Kotronis et al. [101] extend SysML to integrate cost analysis into model-based systems engineering, enabling the evaluation of design alternatives under specific cost and performance restrictions. This integration showcases the potential for future work in automating COSMIC measurement using various UML or SysML artifacts.

While the related work has made significant strides in automating COSMIC measurement, several key differences exist between these approaches and CosMet. Firstly, unlike existing approaches that rely heavily on specific artifacts such as UML diagrams, structured requirements, source code, and more, CosMet is designed to process natural language requirements, offering flexibility and early-stage software development applicability when such artifacts are often absent. Secondly, many existing approaches require manual intervention or source code modification to function effectively. Conversely, CosMet is designed to be fully automated, significantly reducing the manual effort required for COSMIC measurement. Thirdly, while some existing approaches utilize traditional rule-based systems or machine learning models, CosMet leverages advanced LLMs for its operation, allowing CosMet to handle the ambiguity and complexity of natural language requirements more effectively. Lastly, CosMet measures the functional size and provides a transparent and detailed analysis of the COSMIC concepts applied to the functional requirements. This feature is not commonly found in the existing approaches, making CosMet a valuable approach for interpreting the COSMIC measurement process.

8 CONCLUSION

We have presented an LLM-based approach, named CosMet, to measure use cases specified in natural language. We instantiated CosMet using GPT-4. However, our approach allows flexibility in choosing the most suitable LLM for parsing language in use case specifications and producing COSMIC measurement analysis. We evaluated CosMet with an empirical study comprising 123 use cases and 119 functional processes. Seven professional measurers assessed the results and evaluated the tool from a quantitative and a qualitative point of view. The results of our study have revealed that CosMet is effective, transparent, and efficient in automating COSMIC measurement of use case models. CosMet provides a transparent and interpretable analysis of use cases, allowing practitioners to understand how the measurement is derived and to make any necessary adjustments. This transparency enables practitioners to fine-tune the measurement and address any ambiguities or discrepancies in the use case requirements. It also improves the accuracy and reliability of the measurement process and ensures that the measurement aligns with the specific context and requirements of the software project. In addition, CosMet surpasses current NLP approaches in COSMIC measurement for effectiveness and completeness.

Thus, our solution has the potential to bring significant advancements in the way FSM is performed in the software industry, making it more accurate, efficient, and cost-effective. Professionals may use the CosMet-generated output to estimate the functional size measurement or refine it further. However, we identified a few limitations of our approach, precisely the limited LLM's context size (i.e., GPT-4 has a context size of 8,192 tokens). Nevertheless, given the premises, we believe that companies can adopt our approach in their measurement practices. Indeed, by refining GPT-4 with their datasets and linguistic style, companies can use CosMet to automate the COSMIC measurement. Furthermore, fine-tuning GPT-4 with more use cases may allow us get more out of the model. Therefore, our future work will include the identification of several multilingual datasets to fine-tune GPT-4. Moreover, future work could explore the integration of alternative and supplementary UML or SysML model types or other artifacts into automated COSMIC measurement approaches to enhance the evaluation of design alternatives under specific cost and performance restrictions. It is also important to note that many other size measurement frameworks exist, such as standard function points, use case points, and story points, which could be explored in future research using similar methods. Finally, we also plan to explore domain-specific training to address the sensitivity of LLMs to different discourse domains, ensuring CosMet's robustness and adaptability across various industries.

Acknowledgment

The authors thank Kiranet s.r.l and Innovaway S.p.a. for providing the use case documents. The authors also thank Kiranet for providing the professional measurers and the GUFPI-ISMA and Tommaso Galiano for supporting the measurement of the use case documents. Finally, the authors would like to thank the associated handling editor and the anonymous reviewers for their insightful suggestions and feedback, which were instrumental in improving the quality of our manuscript. This work has been partially supported by the Italian PNRR MUR project PE0000013-FAIR.

References

[1] A. Abran and P. N. Robillard, "Function points analysis: an empirical study of its measurement processes," *IEEE Transactions on Software Engineering*, vol. 22, no. 12, pp. 895–910, 1996.

[2] Y. Cheung, R. Willis, and B. Milne, "Software benchmarks using function point analysis," *Benchmarking: An International Journal*, vol. 6, no. 3, pp. 269–276, 1999.

[3] B. Ozkan, O. Turetken, and O. Demirors, "Software functional size: For cost estimation and more," in *European Conference, EuroSPI 2008*. Springer, 2008, pp. 59–69.

[4] D. Rodríguez, M. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," *J. of Systems and Software*, vol. 85, no. 3, pp. 562–570, 2012.

[5] H. Huijgens, A. Van Deursen, and R. Van Solingen, "The effects of perceived value and stakeholder satisfaction on software project impact," *Inf. and Soft. Technology*, vol. 89, pp. 19–36, 2017.

[6] K. R. Jayakumar and A. Abran, "Estimation models for software functional test effort," *Journal of Software Engineering and Applications*, vol. 10, no. 4, pp. 338–353, 2017.

[7] G. Schneider and J. Winters, "Applied use cases," *A Practical Guide*, 2001.

[8] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.

[9] W. Li and S. Henry, "Object-oriented metrics that predict maintainability," *Journal of systems and software*, vol. 23, no. 2, pp. 111–122, 1993.

[10] B. Henderson-Sellers, "Identifying internal and external characteristics of classes likely to be useful as structural complexity metrics," in *OOIS'94: 1994 International Conference on Object Oriented Information Systems 19–21 December 1994, London*. Springer, 1995, pp. 227–230.

[11] L. A. Laranjeira, "Software size estimation of object-oriented systems," *IEEE Transactions on software engineering*, vol. 16, no. 5, pp. 510–522, 1990.

[12] M. Lorenz and J. Kidd, *Object-oriented software metrics: a practical guide*. Prentice-Hall, Inc., 1994.

[13] R. E. Park *et al.*, *Software size measurement: A framework for counting source statements*. Citeseer, 1992.

[14] A. J. Albrecht, "Measuring application development productivity," in *Proc. joint share, guide, and ibm application development symposium*, 1979, pp. 83–92.

[15] ISO/IEC, "Iso/iec 14143-1: 2007. information technology-software measurement-functional size measurement—part 1: Definition of concepts," 2007.

[16] IFPUG, "International function point users group (ifpug) function point counting practices manual," 2000.

[17] C. R. Symons, *Software sizing and estimating: Mk II FPA (function point analysis)*. John Wiley & Sons, Inc., 1991.

[18] C. Gencel and O. Demirors, "Functional size measurement revisited," *ACM Transactions on Software Engineering and Methodology*, vol. 17, no. 3, pp. 1–36, 2008.

[19] COSMIC, "The cosmic funcional size measurement method, version 5. measurement manual." [Online]. Available: https://cosmic-sizing.org/measurement-manual/

[20] K. Lind and R. Heldal, "A practical approach to size estimation of embedded software components," *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 993–1007, 2012.

[21] C. Gencel, "How to use cosmic functional size in effort estimation models?" in *Int. Conf. on Software Process and Product Measurement*. Springer, 2008, pp. 196–207.

[22] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, "Web effort estimation: function point analysis vs. cosmic," *Information and Software Technology*, vol. 72, pp. 90–109, 2016.

[23] C. Commeyne, A. Abran, and R. Djouab, "Effort estimation with story points and cosmic function points-an industry case study," *Software Measurement News*, vol. 21, no. 1, pp. 25–36, 2016.

[24] S. Abrahão, L. De Marco, F. Ferrucci, J. Gomez, C. Gravino, and F. Sarro, "Definition and evaluation of a cosmic measurement procedure for sizing web applications in a model-driven development environment," *Inf. and Soft. Technology*, vol. 104, pp. 144–161, 2018.

[25] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, "Assessing the effectiveness of approximate functional sizing approaches for effort estimation," *Inf. and Soft. Technology*, vol. 123, p. 106308, 2020.

[26] C. Rolland and C. B. Achour, "Guiding the construction of textual use case specifications," *Data & Knowledge Engineering*, vol. 25, no. 1-2, pp. 125–160, 1998.

[27] A. Cockburn, *Writing effective use cases*. Pearson Education, 2001.

[28] L. Li, "Translating use cases to sequence diagrams," in *Fifteenth IEEE Int. Conf. on Automated Software Engineering*, 2000, pp. 293–296.

[29] H. Soubra, Y. Abufrikha, A. Abran *et al.*, "Towards universal cosmic size measurement automation." in *IWSM-Mensura*, 2020.

[30] G. De Vito, F. Ferrucci, and C. Gravino, "Design and automation of a cosmic measurement procedure based on uml models," *Software and Systems Modeling*, vol. 19, pp. 171–198, 2020.

[31] T. Zaw, S. Z. Hlaing, M. M. Lwin *et al.*, "An automated software size measurement tool based on generation model using cosmic function size measurement," in *Int. Conf. on Advanced Information Technologies*. IEEE, 2019, pp. 268–273.

[32] S. Karim, S. Liawatimena *et al.*, "Automating functional and structural software size measurement based on xml structure

of uml sequence diagram," in *IEEE Int. Conf. on Cybernetics and Computational Intelligence*, 2017, pp. 24–28.

[33] N. Chamkha, A. Sellami, and A. Abran, "Automated cosmic measurement of java swing applications throughout their development life cycle." in *IWSM-Mensura*, 2018, pp. 20–33.

[34] S. Bagriyanik and A. Karahoca, "Automated cosmic function point measurement using a requirements engineering ontology," *Information and Software Technology*, vol. 72, pp. 189–203, 2016.

[35] D. Čeke and B. Milašinović, "Automated web application functional size estimation based on a conceptual model," in *23rd Int. Conf. on Software, Telecommunications and Computer Networks*. IEEE, 2015, pp. 234–241.

[36] R. Gonultas and A. Tarhan, "Run-time calculation of cosmic functional size via automatic installment of measurement code into java business applications," in *Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, 2015, pp. 112–118.

[37] Ö. Eren, B. Özkan, and O. Demirörs, "Pl fsm: An approach and a tool for the application of fsm in spl environments," in *Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, 2015, pp. 119–126.

[38] A. Tarhan and M. A. SAĞ, "Cosmic solver: A tool for functional sizing of java business applications," *Balkan Journal of Electrical and Computer Engineering*, vol. 6, no. 1, pp. 1–8, 2018.

[39] H. Soubra, A. Abran, and A. Ramdane-Cherif, "A refined functional size measurement procedure for real-time embedded software requirements expressed using the simulink model," in *IWSM-Mensura*. IEEE, 2012, pp. 70–77.

[40] H. Soubra, A. Abran, S. Stern, and A. Ramdan-Cherif, "Design of a functional size measurement procedure for real-time embedded software requirements expressed using the simulink model," in *IWSM-Mensura*. IEEE, 2011, pp. 76–85.

[41] A. Oriou, E. Bronca, B. Bouzid *et al.*, "Manage the automotive embedded software development cost & productivity with the automation of a functional size measurement method (cosmic)," in *IWSM-Mensura*. IEEE, 2014, pp. 1–4.

[42] A. Sellami, M. Haoues, and H. Ben-Abdallah, "Automated cosmic-based analysis and consistency verification of uml activity and component diagrams," in *8th Int. Conf., Revised Selected Papers 8*. Springer, 2013, pp. 48–63.

[43] A. A. Akca and A. Tarhan, "Run-time measurement of cosmic functional size for java business applications: Is it worth the cost?" in *IWSM-Mensura*. IEEE, 2013, pp. 54–59.

[44] M. S. Jenner, "5.1 automation of counting of functional size using cosmic ffp in uml," *COSMIC Function Points: Theory and Advanced Practices*, vol. 276, 2016.

[45] S. Azzouz and A. Abran, "A proposed measurement role in the rational unified process and its implementation with iso 19761: Cosmic-ffp," in *Software Measurement European Forum, Italy*, 2004.

[46] M. Jenner, "Cosmic-ffp and uml: Estimation of the size of a system specified in uml–problems of granularity," in *Fourth European Conference Soft. Measurement and ICT Control*, 2001, pp. 173–184.

[47] H. Diab, F. Koukane, M. Frappier, and R. St-Denis, "μcrose: automated measurement of cosmic-ffp for rational rose realtime," *Inf. and Soft. Technology*, vol. 47, no. 3, pp. 151–166, 2005.

[48] K. Lind and R. Heldal, "A model-based and automated approach to size estimation of embedded software components," in *ACM/IEEE The 14th Int. Conf. on Model Driven Engineering Languages and Systems*. Springer, 2011, pp. 334–348.

[49] E. Ungan and O. Demirörs, "A functional software measurement approach to bridge the gap between problem and solution domains," in *Software Measurement*, A. Kobyliński, B. Czarnacka-Chrobot, and J. Świerczek, Eds. Springer, 2015, pp. 176–191.

[50] I. Hussain, O. Ormandjieva, and L. Kosseim, "Mining and clustering textual requirements to measure functional size of software with cosmic." in *Soft. Eng. Research and Practice*, 2009, pp. 599–605.

[51] E. Ungan, C. Hammond, and A. Abran, "Automated cosmic measurement and requirement quality improvement through scopemaster® tool." in *IWSM-Mensura*, 2018, pp. 1–13.

[52] F. Wang, F. Ma, G. Song, and J. Zhang, "An improved cosmic software evaluation method," in *IEEE Conference on Telecommunications, Optics and Computer Science*, 2022, pp. 222–226.

[53] M. Ochodek, "Approximation of cosmic functional size of scenario-based requirements in agile based on syntactic linguistic features—a replication study," in *IWSM-Mensura*. IEEE, 2016, pp. 201–211.

[54] ——, "Functional size approximation based on use-case names," *Information and Software Technology*, vol. 80, pp. 73–88, 2016.

[55] I. Hussain, L. Kosseim, and O. Ormandjieva, "Approximation of cosmic functional size to support early effort estimation in agile," *Data & Knowledge Engineering*, vol. 85, pp. 2–14, 2013.

[56] ——, "Towards approximating cosmic functional size from user requirements in agile development processes using text mining," in *Int. Conf. on Applications of Natural Language to Information Systems*. Springer, 2010, pp. 80–91.

[57] S. Sarsa, P. Denny, A. Hellas, and J. Leinonen, "Automatic generation of programming exercises and code explanations using large language models," in *ACM Conference on International Computing Education Research-Volume 1*, 2022, pp. 27–43.

[58] D. Sobania, M. Briesch, C. Hanna, and J. Petke, "An analysis of the automatic bug fixing performance of chatgpt." *arXiv:2301.08653.*, 2023.

[59] A. Ahmad, M. Waseem, P. Liang *et al.*, "Towards human-bot collaborative software architecting with chatgpt." *arXiv:2302.14600.*, 2023.

[60] OpenAI, "Gpt-4 technical report," *arXiv:2303.08774*, 2023.

[61] Anthropic. (2024) Claude 3.5 sonnet (version 3.5). [Online; accessed 15 November 2024]. [Online]. Available: https://claude.ai/

[62] H. Touvron, T. Lavril, G. Izacard, X. Martinet *et al.*, "Llama: Open and efficient foundation language models," *arXiv:2302.13971*, 2023.

[63] J. Wei, X. Wang, D. Schuurmans *et al.*, "Chain of thought prompting elicits reasoning in large language models." *arXiv:2201.11903*, 2022.

[64] J. Wei, Y. Tay, R. Bommasani *et al.*, "Emergent abilities of large language models." *arXiv:2206.07682*, 2022.

[65] E. M. Bender, T. Gebru *et al.*, "On the dangers of stochastic parrots: Can language models be too big?" in *ACM Conference on Fairness, Accountability, and Transparency*, 2021, p. 610–623.

[66] Y. Lu, M. Bartolo, A. Moore *et al.*, "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity," *arXiv:2104.08786*, 2021.

[67] M. Chen, J. Tworek, H. Jun, and et al., "Evaluating large language models trained on code," *arXiv:2107.03374*, 2021.

[68] E. Kasneci, K. Sessler *et al.*, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[69] W. X. Zhao, K. Zhou, J. Li, T. Tang, and et al., "A survey of large language models," *arXiv:2303.18223*, 2023.

[70] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need." in *Advances in neural inf. processing systems*, vol. 30, 2017.

[71] T. Brown, B. Mann, N. Ryder, M. Subbiah *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. L. et al., Ed., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[72] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding." in *naacL-HLT*, vol. 1, 2019, p. 2.

[73] H. W. Chung, L. Hou, S. Longpre, B. Zoph *et al.*, "Scaling instruction-finetuned language models," *arXiv:2210.11416*, 2022.

[74] I. A. Zahid, S. S. Joudar, A. Albahri, O. Albahri, A. Alamoodi, J. Santamaría, and L. Alzubaidi, "Unmasking large language models by means of openai gpt-4 and google ai: A deep instruction-based analysis," *Intelligent Systems with Applications*, vol. 23, p. 200431, 2024.

[75] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, "Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design," in *Generative AI for Effective Software Development*. Springer, 2024, pp. 71–108.

[76] A. Nguyen-Duc, "Generative ai for effective software development."

[77] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm." in *Conference on Human Factors in Computing Systems*, 2021, pp. 1–7.

[78] D. Yogatama, C. D. M. d'Autume *et al.*, "Learning and evaluating general linguistic intelligence." *arXiv:1901.11373*, 2019.

[79] T. Linzen, "How can we accelerate progress towards human-like linguistic generalization?" *arXiv:2005.00955*, 2020.

[80] T. Schick and H. Schütze, "Exploiting cloze questions for few shot text classification and natural language inference." *arXiv:2001.07676*, 2020.

[81] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners." *arXiv:2012.15723*, 2020.

[82] T. Bansal, R. Jha, and A. McCallum, "Learning to few-shot learn across diverse natural language classification tasks." *arXiv:1911.03863*, 2019.

[83] OpenAI, "Prompt engineering guide," https://platform.openai.com/docs/guides/prompt-engineering, 2023, accessed: 2023-10-17.

[84] S. Ekin, "Prompt Engineering For ChatGPT: A Quick Guide To Techniques, Tips, And Best Practices," 2023. [Online]. Available: https://www.techrxiv.org/articles/preprint/Prompt_Engineering_For_ChatGPT_A_Quick_Guide_To_Techniques_Tips_And_Best_Practices/22683919

[85] G. De Vito, S. Di Martino, F. Ferrucci, C. Gravino, and F. Palomba, "Comet github repository." [Online]. Available: https://github.com/gadevito/COSMET

[86] COSMIC, "Course registration ('c-reg') system case study." [Online]. Available: https://cosmic-sizing.org/wp-content/uploads/2020/04/C-Reg-Case-Study

[87] A. Holtzman, J. Buys, L. Du *et al.*, "The curious case of neural text degeneration," *arXiv:1904.09751*, 2019.

[88] F. Pedregosa, G. Varoquaux *et al.*, "Scikit-learn: Machine learning in python." *the Journal of machine Learning research*, 2011.

[89] C.-Y. Lin and F. Och, "Looking for a few good metrics: Rouge and its evaluation," in *Ntcir workshop*, 2004.

[90] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *ACL*, 2004.

[91] X. Franch, C. Palomares, C. Quer, P. Chatzipetrou, and T. Gorschek, "The state-of-practice in requirements specification: an extended interview study at 12 companies," *Requirements Engineering*, vol. 28, no. 3, pp. 377–409, 2023.

[92] COSMIC, "Rise cooker v.2.0.1 case study." [Online]. Available: https://cosmic-sizing.org/wp-content/uploads/2018/08/Rice-Cooker-2.0.1-GoogleDocs.pdf

[93] ——, "Automatic line switching v.1.1." [Online]. Available: https://cosmic-sizing.org/publications/automatic-line-switching-v1-1/

[94] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[95] K. Papineni, S. Roukos *et al.*, "Bleu: a method for automatic evaluation of machine translation," in *Annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[96] M. Ochodek, S. Kopczyńska, and M. Staron, "Deep learning model for end-to-end approximation of cosmic functional size based on use-case names," *Information and Software Technology*, vol. 123, p. 106310, 2020.

[97] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv preprint arXiv:1904.09675*, 2019.

[98] M. Ecar, F. N. Kepler, and J. P. S. da Silva, "Autocosmic: Cosmic automated estimation and management tool," in *Brazilian Symposium on Information Systems*, 2018, pp. 1–8.

[99] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[100] G. De Vito and F. Ferrucci, "Approximate COSMIC size: The quick/early method," in *EUROMICRO Conference on Software Engineering and Advanced Applications, Verona, Italy, August 27-29, 2014*. IEEE, 2014, pp. 69–76.

[101] C. Kotronis, M. Nikolaidou, A. Tsadimas, C. Michalakelis, and D. Anagnostopoulos, "Extending sysml to integrate cost analysis into model-based systems engineering," *IEEE Transactions on Engineering Management*, vol. 71, pp. 2865–2880, 2022.

Gabriele De Vito received a master's degree in computer science from the University of Salerno, where is currently attending the Ph.D. program in computer science. He worked as CTO at various Italian IT firms, and now serves as a High School Teacher and professional consultant since 2021. His certifications include COSMIC, Function Point Specialist, and PRINCE2. He served on program committees for international conferences (e.g., EASE) and reviews for journals in software engineering (eg., IEEE TSE, Springer ESE). He has co-chaired Mensura 2022/2023. His research interests lie primarily in software engineering, machine learning, and large language models.

Sergio Di Martino is a Full Professor of Computer Science at the Department of Electrical Engineering and Information Technology of the University of Naples Federico II, Italy, where he is also the co-chair of the Knowledge Management and Engineering (Knome) Lab. He has published more than 150 papers in journals, conference proceedings and books. His research interests focus on management and analytics of complex and massive datasets, software engineering and software architectures.

Filomena Ferrucci is a Full Professor of Computer Science at the University of Salerno, Italy. She is co-director of the SQM/Web Engineering Lab and a member of the SeSa Lab. Her research focuses on Empirical Software Engineering, Software Metrics, Machine Learning, and Search-Based Software Engineering. She has co-authored over 200 peer-reviewed publications, co-edited three books, and guest-edited two special issues. She coordinates the Ph.D. program in Computer Science, serves as program co-chair of the International School on Software Engineering, and is a member of the Editorial Board of IEEE Transactions on Software Engineering.

Carmine Gravino is a Full Professor of Computer Science at University of Salerno, co-director of SQM/Web Engineering Lab and member of SeSa Lab. Software Engineering expert specializing in visual language modeling, ML-based defect prediction, software maintenance, and AI applications. Current research focuses on methodologies for managing non-functional requirements including privacy, security, and fairness. Published 100+ papers in journals and conferences. He reviews for IEEE TSE, Springer ESEJ, and serves on program committees for conferences like ESEM and IEEE ICSME. He serves on the SEAA Steering Committee, co-chaired several conferences (SEAA 2016, Mensura 2022/2023, Wails 2024), and is on editorial boards for Journal of Software: Evolution and Process and Springer's Software Quality Journal.

Fabio Palomba is an Associate Professor of Computer Science at the University of Salerno and member of the Software Engineering (SeSa) Lab. His research focuses on software maintenance and evaluation, empirical software engineering, and source code quality. He has been recipient of various Distinguished Paper and Best Paper Awards. He serves on program committees for major conferences and as a referee for key journals in software engineering. He has been program co-chair for ICPC 2021 and SANER 2024, and has chaired various tracks and workshops. He serves on editorial boards for several software engineering journals, earning 14 Distinguished/Outstanding Reviewer Awards.