Sizing Software Application Generators: A Case Study of the SimSE Educational Games Generator for Software Engineering Students

Ling Guo[1], Sion Israel Sion[1], Alain Abran[1], Monica Villavicencio[2] and J.M. Desharnais[1]

[1]*École de technologie supérieure, Department of Software and IT Engineering, Montréal, Canada*

[2]*ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL, Facultad de Ingeniería en Electricidad y Computación, Guayaquil, Ecuador*

Abstract

There is no published evidence that reports the functional size results for both an application generator and a set of generated applications under one consistent measurement rationale. This gap is particularly visible for serious games and educational simulation portfolios, which one generator can produce multiple game instances. Measuring the generator itself is important for maintenance and evolution planning, because changes in the generator can affect all generated applications. This paper reports on the functional sizing of an application generator and a set of generated applications. Using COSMIC function points (CFP) – ISO 19761 – this study sized the functional user requirements of the 2024 SimSE educational game portfolio, including its (i) game generator, (ii) shared analyzer tool, and (iii) six generated educational games. Overall, the size of the 2024 SimSE portfolio reaches 853 CFP. These results provide an empirical baseline for the functional sizing of educational game generators and their products, and support future work on CFP-based effort estimation and evolution planning for SimSE-like portfolios as well as replication studies.

Keywords

COSMIC, ISO 19761, Function Points, educational games, serious games, applications generator

1. Introduction

A software application generator is a tool that generates software applications using a description of the desired functionality. This paradigm is often summarized as "software produced by the software" and is also referred to as software-generated or automatically generated software. In practice, a generator receives structured inputs (e.g., models, rules, templates, and configuration data) that represent the required behavior of the target application and produce executable artifacts. In some contexts, developers can also extend the generated application with additional programming. There are two kinds of generated software: (i) generated software that functions independent of its generator (e.g., games created with Game Maker) [1], and (ii) generated software that functions only in combination with its generator or platform (e.g., the educational games of SimSE, which rely on a shared simulation environment and engine) [14, 15, 16]

Functional size measurement (FSM) is widely used in software project management because it provides an objective size input for effort estimation models and supports measurement-based planning and control [1]. Recent COSMIC-based empirical studies illustrate its use across domains, including web analytics implementations [2], healthcare mobile apps [3], and hybrid classical-quantum software systems [4], as well as emerging extensions such as Q-COSMIC for quantum software measurement. [5]. Among the ISO-recognized FSM methods (IFPUG (ISO 20926), NESMA (ISO 24570) and COSMIC (ISO 19761)), COSMIC is widely regarded as a second-generation FSM method grounded in software engineering principles [6]. It has been applied in a variety of domains, including business applications, real-time software, and multiple technology contexts [7, 6]. It also enables ongoing monitoring through size-based delivery indicators (e.g., delivered function points per iteration or release, effort per function

IWSM/MENSURA 2026: International Workshop on Software Measurement and MENSURA Conference, 2026

ling.guo@etsmtl.ca (L. Guo); israel.sion@etsmtl.ca (S. I. Sion); alain.abran@etsmtl.ca (A. Abran); mvillavi@espol.edu.ec (M. Villavicencio); jean-marc.desharnais@etsmtl.ca (J.M. Desharnais)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

point, and defects per function point). The FSM also supports benchmarking of productivity and cost efficiency against internal and external repositories [8, 9, 10, 11]. Therefore, the FSM can act as a common denominator for planning, controlling, and comparing projects and organizational units.

Functional Size Measurement (FSM) is attractive for generator-centric portfolios because it measures software size based on functional user requirements and is intended to be technology-independent. However, software application generators raise a measurement challenge: the same "intended functionality" is represented partly as data entered into the generator (to describe what should be generated) and partly as the behavior of the generated application once it executes. This can lead to inconsistent scoping and double counting when the measurement boundary is not explicit. Template-based model-to-text transformation approaches also characterize generators as systems that translate structured specifications into executable artifacts [12]. This generator-based development motivates a portfolio-oriented view: one generator can support multiple generated applications, which introduces new needs for systematic measurement and estimation.

In 2024, the COSMIC community published a dedicated sizing guideline for "generating applications" and illustrated it with a game generator (e.g., Game Maker), where a game can be produced without writing code, or with optional extra code when needed [13]. The COSMIC guideline addresses this by clarifying the role of the functional user, triggering events, and decomposition of generated-application size into (i) the size of the entered data, (ii) the size of the outputs required by the developer (when applicable), and (iii) the size of any functionality implemented via additional programming (when applicable) [13].

However, there is no published evidence that reports the functional size results for both an application generator and set of generated applications under one consistent measurement rationale. This gap is particularly visible for serious games and educational simulation portfolios, in which one generator can produce multiple game instances.

This study reports on the sizing of functional requirements implemented in the 2024 version of the SimSE portfolio, a software engineering simulation environment that supports the creation of multiple software process simulation games from different models (for example, SimSE environment supports a family of generated game instances) [14, 15, 16]. Specifically, this study reports the COSMIC functional size of the SimSE generator, its shared analyzer tool, and its six generated educational games. A consistent measurement strategy across the games generator, its generated games, and a shared analyzer tool is adopted because these software components are closely related but represent different functional perspectives, which consists of the following parameters, as documented in COSMIC ISO 19761 [7]:

- **Purpose:** To establish baseline functional-size data for the games generator, its shared analyzer tool, and its generated games to support future effort estimation, evolution planning, benchmarking, and replication studies.
- **Scope:** The measurement focuses on functional processes from the instructor/model creator's perspective.
- **Functional users:**
  - Games Generator:The instructor/model creator.
  - Generated games and analyzer tool: The player/student.
- **Granularity level:** Functional processes identified for the instructor/model creator, and the player/student.
- **Layer:** All components within the measurement scope (generator, generated games and analyzer tool) reside in this single application layer.

The rest of this paper is structured as follows. Section 2 presents related work. Section 3 describes the SimSE environment. Section 4 presents the sizing procedure used in this study. Section 5 presents the SimSE portfolio's functional size. Section 6 presents a discussion, validity threats, and future work. The conclusions are presented in Section 7.

2. Related Work

Recent work has intensified efforts to make functional sizing more scalable and repeatable through automation. Moulla et al. proposed an automated COSMIC functional sizing approach from C source code using regular expressions, aiming to reduce manual effort and measurement subjectivity [17]. Lavazza et al. investigated machine learning models for estimating functional size, supporting effort estimation when complete specifications are not yet available [18]. Language-model-based approaches have also been proposed to automate size measurements based on textual requirements. Ünlü et al. reported industrial case studies that automate size measurement from requirements using BERT-style models for COSMIC and MicroM methods [19]. De Vito et al. proposed an LLM-based approach to derive the COSMIC functional size from natural language use cases, with empirical evaluation against expert measurements [20]. These advances strengthen the feasibility of objective sizing in industrial settings; however, they do not resolve the generator-versus-generated boundary problem. They also do not yet provide many empirical baselines for portfolios that explicitly separate generator sizes from generated application sizes.

Application generators are also present in the educational game space, where the "generator plus the generated products" structure is explicit. Anyango and Suleman provided empirical evidence for the suitability of a prototype game generator intended to help instructors create games more easily [21]. SimSE is a relevant portfolio example in software engineering education that provides a creation and simulation environment that supports the creation of multiple software process simulation games from different models (e.g., one environment supports a family of generated game instances) [15, 16]. While the SimSE literature emphasizes educational goals and classroom use, it also highlights the existence of a generator-like tool chain that enables multiple generated applications, which is precisely the context where portfolio-level sizing can support engineering decisions (estimation, evolution planning, and benchmarking).

However, published empirical baselines that report quantitative, engineering-centered measurements of serious-game software systems are scarce, particularly for generators that can produce multiple games. This study addresses this gap by reporting the COSMIC functional size results for the SimSE game generator, several generated educational games, and a shared analyzer tool.

3. Description of SimSE Environment

3.1. General Architecture

SimSE is organized around a generator-based architecture that separates the model construction from the simulation execution. The overall structure in Figure 1 illustrates how an instructor-defined model is transformed into an executable simulation environment used by students.

The architecture can be understood as two main functional subsystems connected through a generation mechanism.

- On the left side: the Creation Environment is responsible for defining the simulation model and producing the corresponding executable simulation.
- On the right-hand side, the Simulation Environment executes the generated model and provides interactive gameplay and analysis capabilities.

Between these two subsystems, the generator component reads the model specification and produces executable artifacts that instantiate the simulation.

This organization establishes a clear boundary between specifications and execution. The model defines the structural and behavioral characteristics of the game, whereas the runtime environment implements and enforces these characteristics during the simulation.

At the end of 2024, the SimSE portfolio consists of a game generator designed to build specific software engineering educational games and six generated educational applications. These six applications correspond to distinct software engineering activities:

- a simulation of a requirements document inspection process,
- a simulation of the Extreme Programming (XP) development process,
- a simulation of the Rational Unified Process (RUP),
- a simulation of the incremental development lifecycle,
- a simulation of the prototyping phase, and
- a simulation of the waterfall development lifecycle.

Each of these games is generated from a dedicated model specification, while relying on the same underlying execution infrastructure.

3.2. Creation Environment

The Creation Environment comprises components responsible for defining and producing executable simulation instances. It includes both the Model Builder and the Generator, which are implemented within the same code repository and operate as a unified subsystem.

The Model Builder provides a graphical interface through which an instructor specifies the structural and behavioral characteristics of a simulation. The underlying textual representation of the specification language remains hidden from users. Instead, the instructor defined the simulation in terms of object types, start state, actions, rules, and graphical configurations. The object types describe the entities involved in the simulation and their attributes. The initial state establishes the initial configuration of these entities. Actions represent possible events occurring during gameplay, while rules define the conditions and effects governing system behavior. Graphical definitions determine how objects and interactions are visually represented in the virtual environment.

The resulting specification was stored as a model-definition file (MDL). This file constitutes a formal representation of a specific simulation scenario and serves as the input to the generator component. Each MDL file corresponds to a distinct simulation scenario and encapsulates the behavioral logic that is executed later in the Simulation Environment.

3.3. Simulation Environment

The Simulation Environment represents the execution subsystem generated from a specified MDL specification. It is an interactive application that students engage in.

At runtime, the environment includes a simulation engine responsible for coordinating the system behavior. A state management component maintains the values of all object attributes, whereas a rule

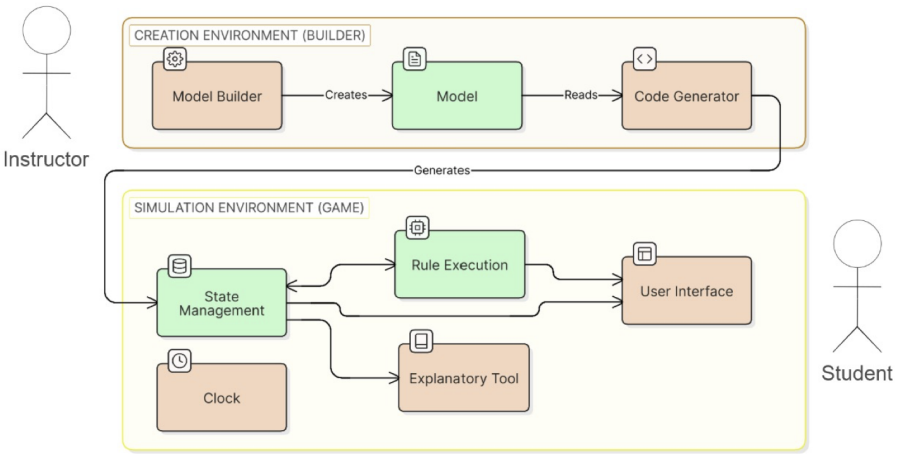


Figure 1: Overall architecture of SimSE. Adapted from [14].

execution mechanism evaluates the defined conditions and applies the state transitions. A simulation clock governs the temporal progression and synchronizes system updates. The graphical user interface provides access to actions and visual feedback to students. An explanatory tool records and displays information about the simulation behavior, supporting the post-game analysis.

Although multiple simulations can be generated from different models, the execution infrastructure remains structurally consistent. Therefore, the runtime system combines stable engine components with model-specific behavioral logic derived from the MDL specification.

3.4. Data Groups

Beyond its architectural organization, SimSE relies on a set of structured data entities that define information manipulated throughout the environment. These entities, referred to as Data Groups in this paper, represent coherent collections of related attributes that describe simulation objects and system states.

At the specification level, Data Groups emerge from the definition of the object types within the Model Builder. Each object type, such as employees, artifacts, tools, or projects, encapsulates a structured set of attributes that collectively characterize an entity of the simulation domain. These structured entities form the basis of the model definition and determine the data elements that are later instantiated at runtime.

At the execution level, the corresponding Data Groups were instantiated and managed by the Simulation Environment. The state management component maintains the current values of their attributes, whereas the rule execution mechanisms read and update these values during the simulation progression. Consequently, Data Groups exist both as structural definitions in the model specification and as dynamic instances within the runtime environment (see [14] for the data model of the relations of the data group classes).

4. COSMIC Sizing Procedure

To size the SimSE portfolio, the following COSMIC FSM steps were followed:

1. Determination of the measurement purpose and measurement scope.
2. Identification of the Functional Processes: This began by analyzing the SimSE game in execution and the available documentation.
3. SimSE cartography was created by identifying the main functional user requirements (FUR), their related functional processes (FPs), and the data groups modified according to ISO 19761 and COSMIC principles [7].
4. Identification of classes: The SimSE data model described in [14] was used to identify the data groups.
5. Mapping Classes to Functional Processes: Associated classes were identified for each functional process.
6. Data Movement Analysis: For each functional process and its associated classes, an analysis of the candidate COSMIC data movements (Entry, Read, Write, Exit):
    - **Entry:** User inputs or data coming into the system.
    - **Read:** Retrieval of data from persistent storage.
    - **Write:** Updating or creating data in persistent storage.
    - **Exit:** Data leaving the system or being displayed to the user.
7. Summation: The total size in terms of the number of COSMIC Function Points (CFP) was obtained by summing all data movements across all associated classes for each functional process.

The purpose of this COSMIC FSM procedure is to measure the SimSE portfolio. The scope comprises a game generator, six educational game models, and an analyzer tool.

These COSMIC measurements were performed in three stages:

(1) Measurement of all the FUR of the SimSE games generator itself.
(2) Measurement of the SimSE six educational models.
(3) Measurement of the Games Analyzer tool.

4.1. Functional User Requirements (FUR) of the Games Generator

Table 1 lists the set of eight functional user requirements (FUR) identified in this game generator, together with their corresponding functional processes. For example, in Table 1, FUR1 includes six functional processes, FUR2 eight functional processes, and so on. In total, the SimSE game generator has 64 functional processes for its set of eight FUR.

Table 1: SimSE Games Generator – 8 FUR with 64 Functional Processes

FUR Id.	**Functional Processes within a FUR**
FUR1: Model Initialization	FP1-1 – File New: Initializes an empty model structure with default parameters. FP1-2 – File Open: Loads an existing model from a file into the application. FP1-3 – File Close: Closes the current model, prompting to save changes if necessary. FP1-4 – File Save: Saves the current model to its existing file location. FP1-5 – File Save As: Saves the current model to a new file location specified by the user. FP1-6 – Exit: Closes the application, prompting to save any unsaved changes.
FUR2: Object Type Definition	FP2-1 – Define New Object Type: Creates a new object type with default attributes. FP2-2 – Add Attribute: Adds a new attribute to an existing object type. FP2-3 – Edit Attribute: Modifies the properties of an existing attribute. FP2-4 – Remove Attribute: Deletes an attribute from an object type. FP2-5 – Rename Object Type: Changes the name of an existing object type. FP2-6 – Remove Object Type: Deletes an object type from the model. FP2-7 – List Object Types: Displays all defined object types in the model.
FUR3: Initial State Configuration	FP3-1 – Start State Modify: Edits the starting values for object instances in the initial state.
FUR4: Action Specification	This function defines the possible events and activities within the simulation: FP4-1 – Create Actions: Defines a new action in the model. FP4-2 – Remove Actions: Deletes an existing action from the model. FP4-3 – Rename Actions: Changes the name of an existing action. FP4-4 – Move Up/Down Actions: Changes the order of actions in the model. FP4-5 – Add Participant: Adds a new participant to an action. FP4-6 – Edit Participant: Modifies the properties of an action participant. FP4-7 – Remove Participant: Deletes a participant from an action. FP4-8 – Move Up/Down Participant: Changes the order of participants within an action. FP4-9 – View Participant Action Options: Displays the configuration options for a participant action. FP4-10 – Edit Participant Action Options: Modifies the configuration options for a participant action.

(continued on next page)

(Table 1 continued)

FUR Id.	Functional Processes within a FUR
FUR5: Rule Definition	FP5-1 – Add New Effect Rule: Creates a new effect rule for an action. FP5-2 – Add New Objects Rule: Defines a rule for creating new objects. FP5-3 – Add New Destroy Objects Rule: Defines a rule for destroying objects. FP5-4 – Configure Rule: Sets up the general properties of a rule. FP5-5 – View Rule: Displays the details of an existing rule. FP5-6 – Edit Rule: Modifies the properties and logic of an existing rule. FP5-7 – Rename Rule: Changes the name of an existing rule. FP5-8 – Remove Rule: Deletes a rule from the model. FP5-9 – View Effects: Displays the effects of a rule. FP5-10 – Edit Effects: Edit the effects of a rule. FP5-11 – View Button Pad: Displays the button pad used for rule creation. FP5-12 – Edit Button Pad: Modifies the button pad used for rule creation. FP5-13 – Add Rule Input Information: Add information for a rule input. FP5-14 – Add Annotation: Adds explanatory notes to a rule.
FUR6: Graphical Configuration	FP6-1 – Graphics Choose an Image: Selects an image to represent an object or action. FP6-2 – View Map: Defines the layout of the simulated environment.
FUR7: Execution Prioritization	FP7-1 – Triggers Prioritizer Update: Modifies the priority order of triggers. FP7-2 – Destroyers Prioritizer Update: Modifies the priority order of destroyers. FP7-3 – Continuous Rules Prioritizer Update: Modifies the priority order of continuous rules. FP7-4 – EstimateProject Trigger Rule Prioritizer Update: Updates priority for estimation triggers. FP7-5 – EndEstimate Trigger Rule Prioritizer Update: Updates priority for end estimation triggers. FP7-6 – Analyze Trigger Rule Prioritizer Update: Updates priority for analysis triggers. FP7-7 – Analyze Destroyer Rule Prioritizer Update: Updates priority for analysis destroyers.
FUR8: Game Generation	FP8-1 – Generate the Model: Compiles the model into a playable simulation game. FP8-2 – File Edit (Icon Directory): Specifies the directory for icon storage. FP8-3 – File Edit (Code Generation Destination Directory): Sets the output directory for generated code. FP8-4 – Model Option Configuration: Configures general options for the model. FP8-5 – View/edit Starting Narrative: Defines the introductory text for the simulation. FP8-6 – View Triggers: Displays existing triggers in the model. FP8-7 – Edit Triggers: Modifies properties of existing triggers. FP8-8 – Remove Triggers: Deletes triggers from the model. FP8-9 – Add New Triggers: Creates new triggers in the model. FP8-10 – Add New Destroyer: Creates new destroyers in the model. FP8-11 – View Existing Destroyers: Displays existing destroyers in the model. FP8-12 – Edit Destroyers: Modifies properties of existing destroyers. FP8-13 – View One Destroyer Detail : Displays properties of existing destroyers. FP8-14 – View Action Options: Displays properties of existing destroyers. FP8-15 – Edit Action Options: Modifies properties of existing action options. FP8-16 – View Constraints: Displays existing constraints in the model. FP8-17 – View Checklist: Shows a checklist of model components or steps.

5. The COSMIC Functional Size of the SimSE Portfolio

5.1. Functional Size of the Games Generator

Table 2 presents a detailed example of the COSMIC measurement for FUR1, including data groups, object of interest, data movements and CFP, while the detailed measurement results for the other seven FURs are provided in Appendix. The measurement result consists of 64 SimSE functional processes for a total functional size of 280

COSMIC Function Points (CFP).

Table 2
Functional Size of FUR1 in Game Generator ($N = 6$ Functional Processes)

No	Functional Process	Data Group	Object of Interest	Data Movement	CFP
FUR1-1	File new	File information	File	E	1
		File information	File	R	1
		File information	File	X	1
FUR1-2	File open	File information	File	E	1
		File information	File	R	1
		File information	File	X	1
		E/C message	E/C message	X	1
FUR1-3	File close	File information	File	E	1
		File information	File	W	1
FUR1-4	File save	File information	File	E	1
		File information	File	W	1
FUR1-5	File save as	File information	File	E	1
		File information	File	W	1
FUR1-6	Exit	File information	File	E	1
		File information	File	W	1
	Total				**15**

In this study, the measured functional size of the SimSE game generator was 280 CFP. The number of data movements by type was 64 CFP for entries, 71 CFP for reads, 70 CFP for writes, and 75 CFP for exits.

5.2. Size of the Six Generated Games

The SimSE portfolio included in 2024 six educational games related to the sets of software engineering activities required:

1. An inspection process of a software requirements document.
2. An extreme programming (e.g., XP) development process.
3. A RUP (Rational Unified Process) development process methodology.
4. An incremental development lifecycle.
5. A prototyping development phase.
6. A waterfall development lifecycle.

Table 3 presents the 22 functional processes identified for the inspection game, together with their corresponding number of data movements by data movement type.

The respective functional sizes of all six educational games are presented in Table 4, which varies from 82 CFP for the smallest inspection game to 91 CFP for the largest waterfall game.

5.3. Functional Size of the Games Analyzer Tool

The Games Analyzer software tool is used for six educational games. It consists of the 11 functional processes listed in Table 5, together with their corresponding numbers of data movements for a total size of 49 CFP.

5.4. Total Functional Size of the SimSE Portfolio

The total functional size of the SimSE educational games portfolio in 2024 consists of the size of the game generator, its six existing educational games, and the analyzer tool for a total of 853 CFP (see Table 6).

In terms of the distribution of functional size across data movement types, it can be observed from Table 6 that:

- The percentages of Entry and Read data movements are similar for the game generator and the set of six game models.
- The percentages of Write and Exit data movements are quite different for the game generator and the set of the six game models.

Table 3
Functional Size of the Inspection Model ($N = 22$ Functional Processes)

No	Functional Process	Entry	Read	Write	Exit	CFP size
1	Graphic with employees	1	1	–	1	3
2	Employee description	1	2	–	2	5
3	Artefacts description	1	2	–	2	5
4	Project description	1	2	–	2	5
5	Choose participant	1	2	1	–	4
6	Choose artefact code	1	2	1	–	4
7	Choose page	1	2	1	–	4
8	Tools description	1	2	–	2	5
9	Next event	1	1	–	1	3
10	Time elapsed	1	1	–	1	3
11	Advance click	1	1	1	1	4
12	No. of ticks	1	1	1	1	4
13	Message (found bugs, etc.)	1	1	1	1	4
14	End (so tired)	1	1	1	1	4
15	Generate object graph	1	1	–	1	3
16	Generate action graph	1	1	–	1	3
17	Composite graph	1	1	–	1	3
18	Trigger rules	1	1	–	1	3
19	Destroy rules	1	1	–	1	3
20	Implementation rules	1	1	1	1	4
21	Actions	1	1	–	1	3
22	Your score	1	1	–	1	3
	Total	**22**	**29**	**8**	**23**	**82**

Table 4
Functional size of the six educational games (2024)

Game Name	Entry	Read	Write	Exit	Total
Inspection	22	29	8	23	82 CFP
XP	24	31	8	25	88 CFP
RUP	24	32	8	26	90 CFP
Incremental	23	30	8	24	85 CFP
Rapid Prototyping	24	31	8	25	88 CFP
Waterfall	24	32	9	26	91 CFP
TOTAL size					**524 CFP**

6. Discussion, Validity Threats, and Future Work

6.1. Discussion

The functional size results reveal a clear portfolio structure for SimSE. The generator accounts for 280 CFP, while the six generated games sum to 524 CFP (82–91 CFP each). The shared Games Analyzer tool adds 49 CFP. In total, the SimSE portfolio has a functional size of 853 CFP.

First, the generator represents a substantial fraction of the overall size. This is consistent with the nature of generating applications. The generator implements generic capabilities that should not be duplicated in each game instance, such as model editing, rule specification, configuration, and code generation. From an engineering perspective, this implies that the generator is a primary long-term software asset. Its evolution can have a system-wide impact, because changes in the generator can affect every generated game. Therefore, sizing the generator separately is useful for planning subsequet development, assessing technical debt, and prioritizing refactoring work.

Second, the generated games are relatively homogeneous in terms of their functional size. The size variation between the smallest and largest games is 9 CFP (82–91 CFP), which is small compared to the generator

Table 5
Games Analyzer Tool – 11 Functional Processes

No.	Functional Process	Entry	Read	Write	Exit	Functional Size
1	Object group	1	1	–	1	3 CFP
2	Show attributes	1	1	–	1	3 CFP
3	Generate object graph	1	2	–	2	5 CFP
4	Action graph	1	1	–	1	3 CFP
5	Generate action graph	1	2	–	2	5 CFP
6	Generate composite graph	1	3	–	3	7 CFP
7	Action (list)	1	1	–	1	3 CFP
8	View trigger rules	1	2	–	2	5 CFP
9	View destroyer rules	1	2	–	2	5 CFP
10	View intermediate rules	1	3	–	3	7 CFP
11	Close	1	1	–	1	3 CFP
	TOTAL	**11**	**19**	**–**	**19**	**49 CFP**

Table 6
Total Functional Size of SimSE Portfolio CFP (2024)

Software Sized	Entry	Read	Write	Exit	Functional Size
Game Generator	64	71	70	75	280 CFP
Six Educational Games	141	185	49	149	524 CFP
Games Analyzer Tool	11	19	–	19	49 CFP
Total Portfolio size					**853 CFP**
Percentage	25%	32%	15%	28%	100%

size. This suggests that the generated games share a stable functional template and differ mainly in terms of parameterization and model content, rather than by distinct functional capabilities. Practically, for planning purposes, the development of a typical SimSE game instance can be approximated by a size near the observed mean (approximately 87 CFP). Such a baseline can support estimation when new models are created, for example, by enabling size-based comparisons between a planned game and an existing game set.

The data movement distribution provided an additional interpretation. Reads and exits are high relative to writes in both the generator and portfolio totals. This aligns with interactive simulation software, where users frequently request information about entities and states and the system frequently presents results and feedback. Persistent updates are less frequent. The analyzer tool also follows this pattern, which is consistent with its role in presenting the simulation information.

From a methodological perspective, this case study illustrates how the COSMIC guidelines for generating applications can be applied to an educational game generator portfolio [13]. A key practical decision is to explicitly define the purpose and scope of the measurement. Treating the generator and each generated game as separate applications improves the traceability and avoids double counting. This makes the results actionable. Educators and maintainers can reason about (i) investment in the generator, (ii) the incremental size of each game instance, and (iii) the shared functionality provided by auxiliary tools. This segmentation is also useful when discussing reuse and amortization. While the generator is costly in terms of size, its overhead can be amortized across multiple games.

6.2. Validity Threats

Several threats to validity should be considered. This was a single-case study of a single generator and a limited set of games. Therefore, the generalization to other educational game generators is limited. In addition, the measurement depends on the completeness and stability of the functional user requirements reconstructed from available artifacts. If undocumented functions exist, functional size could be underestimated. Another potential threat is related to the relatively less explored perspective of measuring the functional size of an application generator. The technical nature of the generator makes its functional user requirements more difficult to reconstruct, which may lead to interpretation challenges. Functional size is also not a direct measure of development effort, and this study did not include effort data. Consequently, the findings should be interpreted as

an engineering baseline rather than an effort model.

6.3. Future Work

Future studies can strengthen this contribution in several directions. First, empirical effort data (e.g., person-hours for generator evolution and for producing new game models) should be collected to calibrate an estimation model based on CFP, the COSMIC units defined in ISO 19761, for sizing functional user requirements. Second, the study can be extended to additional SimSE models or to other simulations and serious-game generators to evaluate external validity. Third, tool support for measurement could be explored, for example, by improving traceability from user interface events and APIs to COSMIC data movements. Finally, combining size data with educational outcomes could help to investigate whether certain functional capabilities or complexity patterns correlate with learning effectiveness in simulation-based SE training.

7. Conclusion

This study provides empirical functional size data for an educational game generator and its products. Using ISO 19761 and the COSMIC Guideline for generating applications, we measured the SimSE game generator (280 CFP), a shared analyzer tool (49 CFP), and six generated educational games (82–91 CFP each, 524 CFP in total). The SimSE portforlio has a functional size of 853CFP.

These results contribute to a quantitative baseline rarely reported in the serious-game literature on SE education. The data can support estimation and maintenance planning for SimSE-like portfolios, and offer a concrete example of applying COSMIC sizing to a generating application. As the next step, we recommend collecting effort data to calibrate CFP-based estimation for generator evolution and game creation, and replicating the approach to other educational game generators to build broader benchmarks.

Acknowledgments

Authors Ling Guo and Sion Israel Sion were supported by grant IT39009 of MITACS (a Canadian federal funding agency), the Philanthropic Research Chair in Software Products and Software Projects at École de technologie supérieure (Canada) and the COSMIC Group.

Declaration on Generative AI

The authors used ChatGPT during the preparation of this manuscript to improve language clarity, fluency, and academic style. The tool was employed for editing and linguistic refinement only; all scientific content, measurement, analyses, and conclusions were developed and validated by the authors. In addition, the AI system Claude (Anthropic) was used to assist us in formating the article according to the template for submission.

References

[1] A. Abran, *Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers*. Hoboken, NJ, USA: Wiley-IEEE Computer Society Press, Apr. 2015, 288 pp., ISBN: 978-1-118-95408-9.

[2] A. Abdallah, A. Abran, M. Qasaimeh, M. Qasaimeh, and B. Abdallah, "Measurement of the Functional Size of Web Analytics Implementation: A COSMIC-Based Case Study Using Machine Learning," *Future Internet*, vol. 17, no. 7, art. 280, 2025, doi: 10.3390/fi17070280.

[3] K. Moumane, L. Sardi, A. Idri, and A. Abran, "Functional size measurement of postnatal care apps: Morocco case study," *Informatics for Health and Social Care*, vol. 49, no. 2, pp. 116–135, 2024, doi: 10.1080/17538157.2024.2332693.

[4] H. Soubra, T. Hacaloglu, A. Abran, and P. Bourque, "Functional Size Measurement of Hybrid Classical-Quantum Software Systems," in *Proc. Global Congress on Emerging Technologies (GCET 2025)*, Lyon, France, Dec. 2–5, 2025.

[5] F. Valdes-Souto, H. G. Perez-Gonzalez, and C. A. Perez-Delgado, "Q-COSMIC: Quantum Software Metrics Based on COSMIC (ISO 19761)," arXiv preprint arXiv:2402.08505, 2024. [Online]. Available: http://arxiv.org/abs/2402.08505. Accessed: 23-Feb-2026.

[6] F. Vogelezang, "Second Generation: Functional Size Measurement by design," COSMIC, Apr. 27, 2022. [Online]. Available: https://cosmic-sizing.org/cosmic-sizing/functional-size-measurement/second-generation/. Accessed: 23-Feb-2026.
[7] ISO 19761:2011, "Software engineering — COSMIC: a functional size measurement method," International Organization for Standardization, Geneva, 2011.
[8] International Software Benchmarking Standards Group (ISBSG), "Functional Size Measurement," Feb. 6, 2025. [Online]. Available: https://www.isbsg.org/2025/02/06/functional-size-measurement/. Accessed: 11-Feb-2026.
[9] Consortium for Information & Software Quality (CISQ), "Software Sizing Standards," 2022. [Online]. Available: https://www.it-cisq.org/standards/software-sizing-standards/. Accessed: 11-Feb-2026.
[10] L. Lavazza, A. Locoro, and R. Meli, "Software Development and Maintenance Effort Estimation Using Function Points and Simpler Functional Measures," *Software*, vol. 3, no. 4, pp. 442–472, 2024, doi: 10.3390/software3040022.
[11] W. Rosa and S. Jardine, "A novel method to early agile effort estimation through functional initiatives," *Journal of Systems and Software*, vol. 223, May 2025, art. 112302, doi: 10.1016/j.jss.2024.112302.
[12] X. He and T. Zan, "BIT: A template-based approach to incremental and bidirectional model-to-text transformation," *Journal of Systems and Software*, vol. 216, art. 112148, Oct. 2024, doi: 10.1016/j.jss.2024.112148.
[13] A. Abran and A. Lesterhuis, "Guideline Measurement of Generating Applications," COSMIC, Version 5.0, Aug. 2024. [Online]. Available: https://cosmic-sizing.org/wp-content/uploads/2024/08/Guideline-Measurement-of-Generating-Applications.pdf. Accessed: 11-Feb-2026.
[14] E. Navarro, "SimSE: A Software Engineering Simulation Environment for Software Process Education," Ph.D. dissertation, Donald Bren School of Information and Computer Sciences, Univ. of California, Irvine, 2006.
[15] E. O. Navarro and A. van der Hoek, "SimSE: An Interactive Simulation Game for Software Engineering Education," *IASTED CATE 2004*, Kauai, HI, USA, Aug. 2004.
[16] E. O. Navarro and A. van der Hoek, "Software Process Modeling for an Interactive, Graphical, Educational Software Engineering Simulation Game," in *International Workshop on Software Process Simulation and Modeling (ProSim)*, 2004.
[17] D. Koulla Moulla, Oumate, E. Mnkandla, H. Soubra, and A. Abran, "Automated COSMIC Function Points Measurement for C Program Using Regular Expressions," *IWSM/MENSURA 2022*, CEUR Workshop Proceedings, vol. 3272, Sep. 2022. [Online]. Available: https://ceur-ws.org/Vol-3272/IWSM-MENSURA22_paper4.pdf. Accessed: 11-Feb-2026.
[18] L. Lavazza, A. Locoro, G. Liu, and R. Meli, "Estimating Software Functional Size via Machine Learning," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 5, art. 114, 2023, doi: 10.1145/3582575.
[19] H. Ünlü, S. Tenekeci, D. E. Kennouche, and O. Demirörs, "Automating software size measurement with language models: Insights from industrial case studies," *Journal of Systems and Software*, vol. 231, art. 112638, Jan. 2026, doi: 10.1016/j.jss.2025.112638.
[20] G. De Vito, S. Di Martino, F. Ferrucci, C. Gravino, and F. Palomba, "LLM-Based Automation of COSMIC Functional Size Measurement From Use Cases," *IEEE Transactions on Software Engineering*, vol. 51, no. 5, pp. 1500–1523, 2025, doi: 10.1109/TSE.2025.3554562.
[21] J. T. Anyango and H. Suleman, "Supporting CS1 Instructors: Design and Evaluation of a Game Generator," *ACM ITiCSE 2021*, pp. 115–121, 2021, doi: 10.1145/3430665.3456306.

A. Detailed COSMIC Measurement Results of FUR2 to FUR8

Table 7
Functional Size of FUR2 in Game Generator ($N = 7$ Functional Processes)

No	Functional Process	Data Group	Object of Interest	Data Movement	CFP
FUR2-1	Define new object type	Object information	Object	E	1
		Object information	Object	R	1
		Object information	Object	W	1
		E/C message	E/C message	X	1
FUR2-2	Add attribute	Attribute information	Attribute	E	1
		Attribute information	Attribute	R	1
		Attribute information	Attribute	W	1
		Attribute information	Attribute	X	1
FUR2-3	Edit attribute	Attribute information	Attribute	E	1
		Attribute information	Attribute	R	1
		Attribute information	Attribute	W	1
		Attribute information	Attribute	X	1
FUR2-4	Rename attribute	Attribute information	Attribute	E	1
		Attribute information	Attribute	R	1
		Attribute information	Attribute	W	1
		Attribute information	Attribute	X	1
FUR2-5	Rename object type	Object type information	Object	E	1
		Object type information	Object	R	1
		Object type information	Object	W	1
		Object type information	Object	X	1
FUR2-6	Remove object type	Object type information	Object	E	1
		Object type information	Object	W	1
		Object type information	Object	X	1
FUR2-7	List object type	Object type information	Object	E	1
		Object type information	Object	R	1
		Object type information	Object	X	1
	Total				**26**

Table 8
Functional Size of FUR3 in Game Generator ($N = 1$ Functional Processes)

No	Functional Process	Data Group	Object of Interest	Data Movement	CFP
FUR3-1	Start State modify	Start State information	Start State	E	1
		Start State information	Start State	R	1
		Start State information	Start State	W	1
		Start State information	Start State	X	1
	Total				**4**

Table 9
Functional Size of FUR4 in Game Generator ($N = 10$ Functional Processes)

No	Functional Process	Data Group	Object of Interest	Data Movement	CFP
FUR4-1	Create actions	Action information	Action	E	1
		Action information	Action	R	1
		Action information	Action	W	1
		Action information	Action	X	1
FUR4-2	Remove actions	Action information	Action	E	1
		Action information	Action	W	1
		Action information	Action	X	1
FUR4-3	Rename actions	Action information	Action	E	1
		Action information	Action	R	1
		Action information	Action	W	1
		Action information	Action	X	1
FUR4-4	Move up/down actions	Action information	Action	E	1
		Action information	Action	W	1
		Updated Action list	Action	X	1
FUR4-5	Add participant	Participant information	Participant	E	1
		Participant information	Participant	R	1
		Participant information	Participant	W	1
		E/C message	E/C message	X	1
FUR4-6	Edit participant	Participant information	Participant	E	1
		Participant information	Participant	R	1
		Participant information	Participant	W	1
		Participant information	Participant	X	1
FUR4-7	Remove participant	Participant information	Participant	E	1
		Participant information	Participant	W	1
		Participant information	Participant	X	1
FUR4-8	Move up/down participant	Participant information	Participant	E	1
		Participant information	Participant	W	1
		Updated Participant information	Participant	X	1
FUR4-9	View participant action options	Action information	Action	E	1
		Action information	Action	R	1
		Action information	Action	X	1
FUR4-10	Edit participant action options	Action information	Action	E	1
		Action information	Action	R	1
		Action information	Action	W	1
		Action information	Action	X	1
	Total				**35**

Table 10
Functional Size of FUR5 in Game Generator (N = 14 Functional Processes)

No	Functional Process	Data Group	Object of Interest	Data Movement	CFP
FUR5-1	Add new effect rule	Effect Rule information	Effect rule	E	1
		Effect Rule information	Effect rule	R	1
		Effect Rule information	Effect rule	W	1
FUR5-2	Add new objects rule	Objects rule information	Objects rule	E	1
		Objects rule information	Objects rule	R	1
		Objects rule information	Objects rule	W	1
FUR5-3	Add new destroy rule	Destroy rule information	Destroy rule	E	1
		Destroy rule information	Destroy rule	R	1
		Destroy objects information	Destroy rule	W	1
FUR5-4	Configure rule	Rule configuration information	Rule	E	1
		Rule configuration information	Rule	W	1
FUR5-5	View rule	Rule information	Rule	E	1
		Rule information	Rule	R	1
		Rule information	Rule	X	1
FUR5-6	Edit rule	Rule information	Rule	E	1
		Rule information	Rule	R	1
		Rule information	Rule	W	1
		Updated Rule information	Rule	X	1
FUR5-7	Rename rule	Rule information	Rule	E	1
		Rule information	Rule	R	1
		Rule information	Rule	W	1
		Updated Rule information	Rule	X	1
FUR5-8	Remove rule	Rule information	Rule	E	1
		Rule information	Rule	W	1
		Updated Rule information	Rule	X	1
FUR5-9	View effects	Effect information	Effect	E	1
		Effect information	Effect	R	1
		Effect information	Effect	X	1
FUR5-10	Edit effects	Effect information	Effect	E	1
		Effect information	Effect	R	1
		Effect information	Effect	W	1
		Effect information	Effect	X	1
FUR5-11	View Button Pad	Button Pad information	Button Pad	E	1
		Button Pad information	Button Pad	R	1
		Button Pad information	Button Pad	X	1
FUR5-12	Edit Button Pad	Button Pad information	Button Pad	E	1
		Button Pad information	Button Pad	R	1
		Button Pad information	Button Pad	W	1
		Button Pad information	Button Pad	X	1
FUR5-13	Add new rule input	Rule input information	Rule	E	1
		Rule input information	Rule	R	1
		Rule input information	Rule	W	1
FUR5-14	Add annotation	Annotation information	Annotation	E	1
		Annotation information	Annotation	W	1
	Total				**44**

Table 11
Functional Size of FUR6 in Game Generator (N = 2 Functional Processes)

No	Functional Process	Data Group	Object of Interest	Data Movement	CFP
FUR6-1	Choose image for graphics	Image information	Image	E	1
		Image information	Image	R	1
		Image information	Image	X	1
FUR6-2	View map	Map information	Map	E	1
		Map information	Map	R	1
		Map information	Map	X	1
	Total				**6**

Table 12
Functional Size of FUR7 in Game Generator ($N = 7$ Functional Processes)

No	Functional Process	Data Group	Object of Interest	Data Movement	CFP
FUR7-1	Triggers prioritizer update	Trigger prioritizer information	Trigger	E	1
		Trigger prioritizer information	Trigger	R	1
		Trigger prioritizer information	Trigger	W	1
		Trigger prioritizer information	Trigger	X	1
FUR7-2	Destroyers prioritizer update	Destroyers prioritizer information	Destroyers	E	1
		Destroyers prioritizer information	Destroyers	R	1
		Destroyers prioritizer information	Destroyers	X	1
		Destroyers prioritizer information	Destroyers	W	1
FUR7-3	Continuous rules prioritizer update	Continuous rules prioritizer list	Continuous rules	E	1
		Continuous rules prioritizer information	Continuous rules	R	1
		Continuous rules prioritizer information	Continuous rules	W	1
		Continuous rules prioritizer information	Continuous rules	X	1
FUR7-4	EPT rule prioritizer update	EPT rule information	EPT rules	E	1
		EPT rule information	EPT rules	R	1
		EPT rule information	EPT rules	W	1
		EPT rule information	EPT rules	X	1
FUR7-5	EET rule prioritizer update	EET rule information	EET rules	E	1
		EET rule information	EET rules	R	1
		EET rule information	EET rules	W	1
		EET rule information	EET rules	X	1
FUR7-6	AT rule prioritizer update	AT rule information	AT rules	E	1
		AT rule information	AT rules	R	1
		AT rule information	AT rules	W	1
		AT rule information	AT rules	X	1
FUR7-7	AD rule prioritizer update	AD rule information	AD rules	E	1
		AD rule information	AD rules	R	1
		AD rule information	AD rules	W	1
		AD rule information	AD rules	X	1
	Total				**28**

Note 1: EPT refers to EstimateProject trigger.
Note 2: EET refers to EndEstimate trigger.
Note 3: AT refers to Analyze trigger.
Note 4: AD refers to Analyze destroyer.

Table 13
Functional Size of FUR8 in Game Generator (N = 17 Functional Processes)

No	Functional Process	Data Group	Object of Interest	Data Movement	CFP
FUR8-1	Generate the model*	Generate instruction	Generating	E	1
		22 DG	22 OOI	R	22
		22 DG	22 OOI	W	22
		22 DG	22 OOI	X	22
FUR8-2	File edit icon directory	File information	File	E	1
		File information	File	R	1
		File information	File	W	1
		E/C message	E/C message	X	1
FUR8-3	File edit code generation	File information	File	E	1
		File information	File	R	1
		File information	File	W	1
		E/C message	E/C message	X	1
FUR8-4	Model option configuration	Model information	Model	E	1
		Model information	Model	W	1
FUR8-5	View/edit Starting narrative	Narrative information	Narrative	E	1
		Narrative information	Narrative	R	1
		Narrative information	Narrative	W	1
		Narrative information	Narrative	X	1
FUR8-6	View triggers	Trigger information	Triggers	E	1
		Trigger information	Triggers	R	1
		Trigger information	Triggers	X	1
FUR8-7	Edit triggers	Trigger information	Triggers	E	1
		Trigger information	Triggers	R	1
		Trigger information	Triggers	X	1
		Trigger information	Triggers	W	1
FUR8-8	Remove triggers	Trigger information	Triggers	E	1
		Trigger information	Triggers	W	1
		Trigger information	Triggers	X	1
FUR8-9	Add new triggers	Trigger information	Triggers	E	1
		Trigger information	Triggers	R	1
		Trigger information	Triggers	W	1
		Trigger information	Triggers	X	1
FUR8-10	Add new destroyers	Destroyers information	Destroyers	E	1
		Destroyers information	Destroyers	R	1
		Destroyers information	Destroyers	W	1
		Destroyers information	Destroyers	X	1
FUR8-11	View existing destroyers	Destroyers information	Destroyers	E	1
		Destroyers information	Destroyers	R	1
		Destroyers information	Destroyers	X	1
FUR8-12	Edit destroyers	Destroyers information	Destroyers	E	1
		Destroyers information	Destroyers	R	1
		Destroyers information	Destroyers	W	1
		Destroyers information	Destroyers	X	1
FUR8-13	View one destroyer detail	Destroyers information	Destroyers	E	1
		Destroyers information	Destroyers	R	1
		Destroyers information	Destroyers	X	1
FUR8-14	View action options	Options information	Options	E	1
		Options information	Options	R	1
		Options information	Options	X	1
FUR8-15	Edit action options	Options information	Options	E	1
		Options information	Options	R	1
		Options information	Options	W	1
		Options information	Options	X	1
FUR8-16	View constraints	Constraints information	Constraints	E	1
		Constraints information	Constraints	R	1
		Constraints information	Constraints	X	1
FUR8-17	View checklist	Checklist information	Checklist	E	1
		Checklist information	Checklist	R	1
		Checklist information	Checklist	X	1
	Total				**122**

Generate the model contains 22 different data groups (generators), which are Engine, ActionGraph, ExplanatoryTool, MultipleTimelinesBrowser, ObjectGraph, AtAGlanceFrames, AtAGlanceTableModel, AttributePanel, SimSEMap, TabPanel, DestroyerChecker, MenuInputManager, MiscUpdater, RuleExecutor, TriggerChecker, ChooseActionToDestroyDialog, ChooseActionToJoinDialog, ChooseRoleToPlayDialog, EmployeeParticipantSelection, NonEmployeeParticipantSelectionDialog, ParticipantSelectionDialogsDriver, Repository.