




Extending SysML to Integrate Cost Analysis Into Model-Based Systems Engineering

Christos Kotronis , Mara Nikolaidou, *Member, IEEE*, Anargyros Tsadimas , Christos Michalakelis ,
and Dimosthenis Anagnostopoulos

Abstract—Model-based systems design (MBSD), a current trend adopted by INCOSE, employs the systems modeling language (SysML), a standard introduced by OMG and INCOSE. Though there are numerous works on integrating performance exploitation in SysML, cost is not sufficiently explored as a driving design parameter. By integrating cost analysis in a popular modeling language like SysML, the proposed approach may be applied to any system designed using standardized languages and tools. In this work, we integrate cost analysis within SysML models at a generic level to explore design alternatives under specific cost and performance restrictions and perform tradeoff analysis. The proposed SysML extensions provide: 1) cost-related entities to encode cost aspects, such as capital and operating expenses, and 2) functions that enable the automatic computation of costs; these extensions are contained in a custom SysML cost profile. The feasibility and benefits of the approach are explored in two distinct real-world case studies with different purpose and characteristics. 1) Configuring a remote elderly monitoring system, taking into consideration patients' budgetary and operational concerns regarding the equipment installed in their homes. In this case, patients had the opportunity to evaluate and prioritize their concerns prior to using the system. 2) Exploring the improvement of the passengers' comfort as a level of service indicator in the Athens Metro railway system, taking into account operational cost constraints. In this case, the operator obtained forecasts of performance and cost of the metro system operation in order to choose between different operational policies.

Index Terms—Capital expenditures (CapEx), cost analysis, model-based system engineering (MBSD), operating expenses (OpEx), railway transportation system, remote elderly monitoring, systems modeling language (SysML).

I. INTRODUCTION

SYSTEMS engineering is a complex activity consisting of designing, operating, and evaluating systems, ranging in scale and complexity, such as a complex public transportation system or a smart city control system. There are numerous methodologies available to address systems engineering,

Manuscript received 7 August 2020; revised 28 July 2021, 11 October 2021, 31 March 2022, and 31 July 2022; accepted 3 August 2022. Date of publication 31 August 2022; date of current version 5 January 2024. This work was supported in part by the HFRI and the GSRT under the HFRI PhD Fellowship grant (GA. no. 1526). Review of this manuscript was arranged by Department Editor E. Kongar. (*Corresponding author: Christos Kotronis.*)

The authors are with the Department of Informatics and Telematics, Harokopio University of Athens, 17671 Kallithea, Greece (e-mail: kotronis@hua.gr; mara@hua.gr; tsadimas@hua.gr; michalak@hua.gr; dimosthe@hua.gr).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TEM.2022.3200148>.

Digital Object Identifier 10.1109/TEM.2022.3200148

while a model-based approach [model-based systems design (MBSD)] is employed in the most recent ones [1]. According to Wymore [2], there are several prominent parameters that should be taken into account when designing a system: besides provided services (system output), available technology and performance, cost is also a crucial factor that constrains design decisions. It should be noted that Wymore adopts the term performance to describe all nonfunctional requirements the system should conform to, also including availability, security, quality of service, etc. In fact, during system design, capital (initial investment), or operating costs drastically affect (and are affected by), as well as constrain the services a system provides to its end-users [3] and should be leveraged against performance metrics, taking into account available technology. This necessity has grown even more during the recent recession, where many systems were redesigned in order to accommodate cost reductions during their operation.

However, popular engineering methodologies [4] are not focusing on exploiting cost parameters while designing a system, since they are typically superseded by performance [5]. In the cases where cost is actually taken into account, not all its different dimensions (e.g., cost categories like operational expenses, equipment costs, etc.) are explored; only the design process costs or the costs incurred during the system's architectural changes (capital costs) are studied [6]. Common pitfalls of related efforts are the lack of generality, scalability, dynamicity, and simplicity, while cost exploitation is customized for a specific category of systems or case studies [7], [8], [9], [10].

An approach to overcome these difficulties should be generic enough to successfully address different system categories and enable cost analysis at both a high-level (e.g., on the level of overall capital or operational expenses of a system) or fine-grained (e.g., the individual cost of a device). In practice, the system designer should be facilitated to model, design, and evaluate a system from a cost perspective as well, and be able to perform both performance and cost analysis of the explored design alternatives in a single modeling environment.

Systems modeling language (SysML) [11] is a standardized modeling language proposed by Object Management Group (OMG) and the International Council on Systems Engineering (INCOSE) for complex systems engineering. It is designed to enable modeling of complex systems of any kind in a hierarchical fashion. It is well accepted by both the academic and industrial community and has been adopted by many MBSD methodologies [12]. There are numerous efforts to enhance the

exploration of system performance as a design parameter, using SysML [13], [14]; however, there is a lack of cost exploitation in a similar manner. To this end, we aim to integrate cost properties and restrictions within SysML models at a generic level, and accommodate the designer to explore design alternatives under specific cost restrictions, e.g., evaluate either acquisition or operating costs for alternative design decisions. This way, we integrate cost analysis in a popular modeling language employed for MBSD, so that it may be explored for any system designed using SysML.

To achieve our goal, we extended available SysML modeling entities in a standardized fashion by constructing a generic SysML profile that encodes cost design parameters. This is a custom cost-related SysML meta-model extension, termed *Cost profile*. To enable computations related to this perspective (e.g., to compute the acquisition and operational expenses of a system), we introduce a cost functions library, which is readily available to the designer during design time, along with the profile. The profile is general and extendable enough to be applied in any system, leveraging the expressiveness and flexibility of SysML. To automate computations, we employ parametric execution [15]. The end results (i.e., the computed cost entity values) are in turn used to populate the parameters of cost profile elements; related requirements (e.g., conformance to budgetary constraints) are then verified by comparing the computed and desired values.

To evaluate the applicability of the approach, we apply the proposed SysML cost-related extensions in two real-world use cases with different scopes: 1) in a healthcare remote elderly monitoring system (REMS) [16], [17] we explored different solutions for the system installed in patients' homes to monitor their health under different service and budget restrictions set by themselves, according to their needs. Capital cost is mainly explored in this case study, which was performed as part of a research project targeting the application of internet of medical things (IoMT) [18], 2) in the railway transportation system (RTS) of Athens Metro [19] we assisted the company operating the system to study the improvement of passenger comfort in stations and trains, measured based on international standards [level-of-service (LoS)] [20] against the corresponding increase of operational cost.

The main contribution of the proposed approach is the seamless integration of cost analysis into SysML in a general fashion, so that it may be applied into any system or system-of-systems modeled with this language. This way both performance and cost constraints may be explored by the system designer while evaluating alternative design solutions. Ideally, a designer should minimize or reduce the costs of the provided system services, while maintaining their performance for greater satisfaction and usage by the end-users.

The rest of this article is organized as follows. First, we present an overview of the related work in Section II. Section III contains a description of proposed SysML extensions and the SysML cost profile, emphasizing on the description, estimation and evaluation of costs. Section IV shows the applicability of the developed profile to 1) the REMS (see Section IV-A) and 2)

the Athens Metro RTS (see Section IV-B). Finally, Section VI concludes this article.

II. BACKGROUND & RELATED WORK

Besides the desired output and available technology (e.g., services), performance and cost are considered critical design parameters that should be measured, evaluated, and preserved during system design [2]. Most system design methodologies, model-based or not, mainly focus on the achievement of diverse performance requirements for the studied system(s) [21], [22]. However, systems engineering has been expanding beyond its original confines, incorporating social and organizational issues, making system design far more complex and challenging [23], [24]. Recognizing and addressing these issues from a systems engineering standpoint is a necessity [25], while integrating a cost perspective is a first step toward this direction. Ignoring incurred costs or other concerns related to system viability results in poorly designed systems.

A. Estimating Cost in MBSD

A number of methodologies are available in the pertinent literature, which perform cost analysis on individual system domains. In the public transportation domain, Gattuso et al. [7] performed a comparison of different cost assessment methodologies with their mathematical formulations. Essentially, they provide transportation planners and policy makers with a systematic process for estimating costs that are representative of the area and service, for analysis and decision-making purposes. Their approach can be used as an intermediate tool to allow planners to more easily perform railroad cost analysis, determining cost functions. In a similar context, in [8] the activity based costing (ABC) is employed as a suitable method for a business-economic transportation cost model, while traditional book-keeping approaches like traditional cost accounting that make rigid assumptions about system resource utilization in railways are deprecated. However, adequate model-based design methods, which are driven by representative system models that properly model and integrate critical design requirements (including costs) are still needed [26]. In our work, we develop a general and customizable cost analysis and evaluation methodology to fill this gap, based on MBSD, and employ public RTS as one of our use cases.

Moving beyond a single application domain and employing MBSD, we identify the following lines of related work. In [9], three methods of estimating system costs: analytical, analogous, and parametric, are described. Focusing mostly on parametric estimates, it proposes a commercial cost model that though has some stringent requirements, such as cost engineer training, fine-grained calibration, and additional components (such as databases). While we also employ parametric execution for cost estimation, our methodology does not have such requirements in order to be effectively used by a system designer/engineer. Constructive systems engineering cost model (COSYSMO) [10] is a well-known costing methodology, widely adopted with model-based system design, targeting systems of systems (SoS).

Most approaches exploit it as basis for effective cost analysis. For example, in [27] the COSYSMO effort equation is employed for a proposed parametric effort formula for constructive cost models. Focus is given on translations between models/tools in the MBSD, specifically mapping architectural elements into behavior/performance analysis and cost model inputs. SysML, Department of Defense Architecture Framework (DoDAF) [28], COSYSMO, constructive cost model (COCOMO) [29], or COCOMO II [10] parametric cost models, and other frameworks and environments are also investigated in that work [27]. As a case study, the RTS for unmanned aerial vehicle intelligence, surveillance, and reconnaissance is explored. In addition, cost model interfaces for components of the architectures are developed in order to evaluate cost effectiveness in such environments. In [30], the aforementioned COSYSMO framework and in particular, a practical implementation of the COSYSMO cost estimating relationship, is exploited. Extending an MBSD modeling environment with SysML, the authors achieve an end-to-end estimation of systems engineering effort in designing and developing a system. In contrast to COSYSMO and related methodologies, we focus on structural and operational costs of the system itself rather than the costs of its design stages. Moreover, while approaches, such as COSYSMO require careful calibration and tuning in order to better predict and estimate costs, our methodology is more straightforward for the designer, and integrates and computes costs directly within the system model.

Spruytte et al. [3] focused on equipment costs which are more relevant to our work (see, e.g., Section IV-A, where we study capital expenses in a healthcare system). Since most studies make use of case-specific *ad hoc* models (typically, a combination of visualization and spreadsheet modeling) that are both hard-to-use and error-prone, the authors develop the equipment coupling modeling notation (ECMN), a flowchart-like notation, based on a small number of building blocks, that facilitates hierarchical modeling by means of nesting models (using sub-models). Instead of ECMN notation, we follow the much more common SysML and expand our modeling methodology also to operational expenditures and related costs (besides capital expenses).

B. UML/SysML Profiles Including Cost Features

Focusing more on SysML, which is widely known and used both in academia and industry [31], we make the following observations. It is appropriate for modeling the structure, behavior, and requirements of domain-specific systems, supporting the accurate, effective, and complete design of a broad range of –complex– systems [12], [32], [33]. To describe domain-specific systems, SysML profiles are used [34], while the *Stereotype* is the basic SysML structural element to define specialized entities based on existing ones, and can be applied to *classes*, *attributes*, and *operations* [35]. According to [36], the main reasons for selecting SysML for system modeling are the following: 1) It has been proven to be expressive enough to model complex systems with heterogeneous components, despite its lack of native

support for cost- or quality-oriented modeling primitives [12], and 2) the SysML profiles provide the necessary flexibility for tuning or extending native constructs; examples include, but are not limited to, generalizations of SysML block or requirement stereotypes [12].

In [5], basic indicators, like cost and performance are explored in the SysML's conceptual framework and an attempt to define guidelines for modeling-related parameters is made. However, that work does not provide solid, practical examples of SysML models and extensions. In addition, Ameller et al. [37] carried out a comprehensive research on managing nonfunctional requirements like cost and performance, through MBSD. They observe that in the design process, several research efforts focus only on specific requirement types and systems. In essence, there is a lack of a comprehensive and generic SysML-based MBSD methodology that adequately covers a broad range of requirements and systems.

C. SysML Cost-Related Profiles

We note that there are more specialized efforts that exploit the SysML profiling mechanism to design a system, while performing a techno-economic or cost analysis to evaluate it [38], [39]. In [40], a tradeoff analysis among alternatives for a system model is presented, addressing MBSD within a SysML context. This way, the authors meet design objectives, such as cost, performance or other inputs derived from the system stakeholders' needs. Since these needs are often conflicting, the tradeoff analysis is expected to provide a balanced solution [12]. Among several methods that are used to specify, design, and verify a system, based on cost or performance factors, the authors propose the scenario-driven object-oriented system engineering method (OOSEM) from OMG that primarily uses SysML. Even though OOSEM can be used to evaluate design objectives such as cost, it reaches a limitation when a large number of system alternatives and solutions must be evaluated, in contrast to our methodology, which integrates the scale of the model seamlessly into the cost profile.

The most relevant effort that also aims for generality is [38], where a model-driven techno-economic approach for the estimation of economic parameters of cloud service deployment is proposed. The authors employ SysML to describe cloud architectures, emphasizing cost-related properties. The total cost of ownership (TCO) for cloud infrastructure and services is their use case; they incorporate TCO into SysML cloud models, while cloud providers are facilitated in computing it. We were motivated by that approach to extend SysML functionality and constructs to integrate cost-related characteristics into SysML models. An evolution of this approach, in the same domain, can be found in [39]; this inspired the use of SysML requirements that are attached to (and should be satisfied by) specific cost entities. However, for both these works it is not clear whether the generality claim holds also for wildly different domain contexts.

III. EXTENDING SysML TO INTEGRATE COST ANALYSIS

A. Overview

In our work, we aim to facilitate generic cost evaluation, enabling different solutions to be explored until the designer can reach a satisfactory system design that remains affordable to its users (in the context of the desired domain). Such an effort entails description and computation of economic parameters, such as the TCO [41], capital expenditures (CapEx), or operational expenses (OpEx) [42], etc. Such parameters are important in/during system design [39] and should thus be explored and estimated, in order to enable a high-level (e.g., on the level of the TCO) or a fine-grained (e.g., the acquisition cost of a single system component) cost analysis of the examined systems.

To understand the challenges of such an analysis, consider the case of a designer who models a system from a cost perspective, employing the widely used SysML modeling language [11]. Without a generic cost-oriented design approach, the designer would have to resort to custom extensions of a domain-specific structural profile [12] of a system by adding cost properties (along with the structural ones) on each and every system component that should be analyzed costwise. This heavily increases complexity, introducing more potential errors (e.g., overlooking a critical cost property), and is neither scalable nor generalizable. Moreover, it would reduce clarity and transparency regarding cost integration, analysis, and estimation, especially when used across multiple system domains.

Drawing an example from the healthcare domain, assume that a system designer is tasked to model a medical monitoring system for every patient room in a 500-room hospital. After selecting the needed components (for example, medical sensor devices and data aggregators), the designer will need to compute the overall CapEx for the hospital. Without integrating this process into the system model and automating it, the designer will have to do this manually for each system component. Moreover, should an upgrade on the equipment take place, the process needs to be performed anew; this is rigid and constrains dynamicity (for example, changing component prices depending on operational properties). It is also important to note that in a heterogeneous system (e.g., a hospital with different medical equipment per room), such a cost analysis becomes untenable without proper automation in place. The same observation applies to other systems/domains. Ideally, the system designer should be able to apply his/her domain expertise without worrying about other cost analysis complexities.

To deal with the aforementioned challenges, we employ and extend SysML to enable the following.

- 1) Domain-independent cost entities (Section III-B).
- 2) Automated (re)computations of costs (Section III-D) incurred by the cost entities.
- 3) Cost computation functions that are readily available in an inventory (Section III-E) to augment the automation features of point 2) in a user-friendly fashion.

We encapsulate these extensions in a comprehensive SysML profile [12], termed *cost profile* (Section III-B). This is a domain-agnostic profile that can adjust to the intricacies of different domains (e.g., different cost categories and design objectives)

and can be combined with other domain-specific profiles. These –combined– profiles are then integrated into SysML system models, making them more accurate and complete.

B. Extended Domain-Independent SysML Cost Elements

It is worth mentioning that the current version of the SysML specification [11], [43] neither directly addresses cost-specific concepts nor provides a standard notation for them, i.e., specific predefined cost SysML elements [12]. However, it provides generic building blocks that can serve as their basis. Thus, we create new cost entities represented as SysML *Blocks*, i.e., SysML's basic modular units for representing entities of a system [12]; these blocks hold cost-specific value properties and correspond to discrete cost categories, e.g., capital, acquisition, operational, etc. Note that, we propose the creation of separate cost entities, instead of overloading a system component as a single element that combines both structural and cost characteristics. Through this decoupling, a clearer, more accurate, and extensible system model can be defined, where the different perspectives, i.e., structural and economical, are distinguished. The designer can specify a system component with its structural properties, and connect it with domain-independent, separate cost entities that hold its cost-specific properties.

These cost entities comprise our SysML cost profile depicted in Fig. 1. Specifically, the illustrated white elements are standard SysML-provided constructs, blue-colored elements are the proposed cost extensions, and grey-colored elements are specializations of these extensions.

The depicted *Cost* element is the center of our profile, defined as the stereotype of the SysML block, and it has the following characteristics.

- i) It contains cost-specific properties, i.e., a *value*, used for the assessment of the system components' worth, a *measUnit* (i.e., measurement unit) that represents the currency (e.g., "Euro"), and an *instances* property that represents the number of occurrences of each system component connected to a cost entity; note that a system may contain multiple instances of the same component. The latter property differentiates from the others since it is derived [12]; the "l" symbol, preceding its name, indicates that the value of this property is produced from another element (here, from the structural components of the system).
- ii) It estimates structural system components since it holds the cost values that characterize them. Conceptually, the components are represented by the *DesignBlock* element, as depicted in Fig. 1; in order to illustrate their connection, there is a custom *estimation* relationship between the cost and the *DesignBlock* element. Note that a system can have a hierarchical structure; a "parent"- "children" hierarchy can be formed by the system components. The same hierarchy applies to the cost entities connected to the structural components.
- iii) It represents a general cost entity that has specific cost specializations; these specializations form hierarchies [as mentioned in point 2)] and inherit all the characteristics of the cost element, as described in Section III-C.

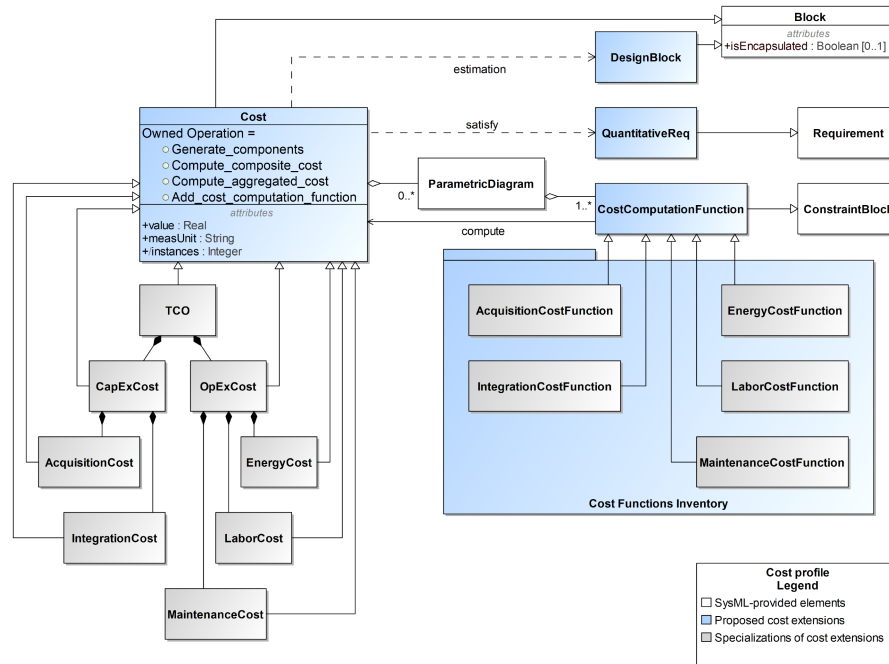


Fig. 1. SysML cost-related extensions.

- iv) It owns specific operations, referring to cost computations. The value of a cost entity can be computed either by automatically summing up the cost values of its children entities, as described in Section III-D or by custom computation functions. Regarding the latter, described in detail in Section III-E, they are represented by the *CostComputationFunction* entity (as shown in Fig. 1) which is the stereotype of SysML's *ConstraintBlock* [12]. Each function holds a specific mathematical expression that generates a cost value. These expressions are computed in a standardized fashion, i.e., via the graphical SysML *Parametric Diagrams* [12], [15]; the system designer inserts the expressions within the diagrams along with appropriate input/output properties, executes them (parametrically), and the computed cost value is extracted for assessment.
- v) It satisfies cost-related quantitative SysML *Requirements*, that specify certain conditions that the system must conform to [12]. This is needed to complete the cost analysis of a system. Specifically, via the *satisfy* relationship (depicted in Fig. 1), the cost entity can meet and fulfill cost-related requirements; e.g., “the cost of component X should not exceed Y Euro.” When the system is actually evaluated (costwise) such requirements are either satisfied or not.

C. Cost-Related Specializations

The first specialization of the cost element (see Fig. 1) is the TCO cost entity that represents the financial assessment and cost accounting tool [41], [44], used to determine the total economic value of an investment, including capital and operating expenses. Typically, organizations and systems have two types of expenses, i.e., CapEx and OpEx [42]. Both cost categories refer to money paid by the organization, although each one is managed differently for accounting or taxation purposes [45].

CapEx is the funds provided by the stakeholders (or end-users) of the system to purchase or improve physical assets, such as system components. OpEx is an ongoing cost, required to keep a system operational on a daily basis. Usually, organizations try to reduce the expenses that are incurred during system operation, without affecting its quality or performance. Typical OpEx examples are energy consumption costs and personnel salaries. We represent CapEx and OpEx categories as *CapExCost* and *OpExCost* specialization entities.

Note that both entities are considered composite, i.e., they comprise different categories of subcost entities. The *CapExCost* contains the *AcquisitionCost* entity that represents the acquisition cost of a newly purchased system component, and the *IntegrationCost* entity, related to the expenses incurred for integrating the acquired components into the system. *OpExCost* is the sum of the *EnergyCost*, associated with the energy consumption costs of the components, the *MaintenanceCost*, related to the expenses incurred to maintain the components operational and in good condition, and the *LaborCost* entity, i.e., the salary of system participants that operate the components and provide services. A system can be analyzed costwise using either or both *CapExCost* and *OpExCost* entities, depending on the system design focus.

D. Cost-Related Computations

It is worth mentioning that we categorize the cost entities into composite, aggregated, and individual, based on the way their values are computed. These entities and their computation process are included in horizontal and vertical tree-like hierarchies, as depicted in Fig. 2. Indicatively, we employ CapEx as a composite cost entity, while its subcost categories, i.e., acquisition and integration, represent aggregated and individual costs.

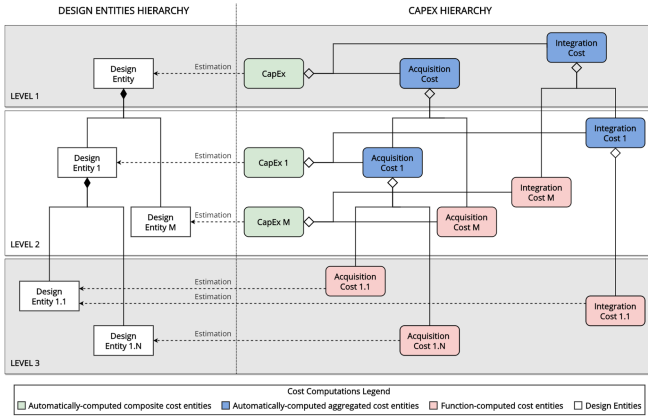


Fig. 2. Automated computation of CapEx.

Traversing the hierarchy from the left to the right side, we have design entities connected to composite cost elements that are linked to aggregated (or individual) cost entity trees. This signifies that each system design entity is estimated by respective composite cost elements (such as TCO, CapEx or OpEx), which in turn are composed of aggregated (or individual) cost elements of different types (e.g., CapEx comprises acquisition and integration cost entities). Note that the estimation (or not) of certain design entities by corresponding composite cost elements can be designated by the designer; that is, certain design entities can be directly estimated by corresponding aggregated or individual cost elements only.

Traversing the hierarchy top-down, we essentially map the structure of the design entities to the corresponding structure of the aggregated and individual cost entities. The computations of the values of the different cost entities comprising this tree start from the leaf nodes and traverse the tree upward, as described in the following.

The leaf nodes are individual cost entities of system components, which may correspond to different cost categories (e.g., the acquisition cost of a system component or its energy consumption cost). The values of these cost entities are computed by cost functions included in an inventory, as described in the following Section III-E. Each individual cost entity is computed by a specific function inserted by the designer.

By summing up the individual –function-computed– costs of a level, the aggregated costs of its parent level (corresponding to more complex design entities) can be computed automatically per type. Specifically, an aggregated cost entity obtains its value by the automatic summation of the values of its children cost entities (corresponding to the respective design entity hierarchy), conforming to the same cost category. For example, the acquisition cost of the part of the system W , comprising the components X and Y , is the summation of the individual acquisition costs of X and Y . This can be done automatically for each level of the hierarchy; note that it is permitted to have a mixture of aggregated and individual costs on a level.

At the composite cost entities of the tree (including the top-level root), the same-level aggregated and individual costs are finally summed up, across the different types, to produce the

Algorithm 1 Computing the Values of Aggregated Cost Entities.

Data: design Entity (dE), cost Entity (cE), cost Type (cT), cost Value (cV)
function computeAggregatedCost (dE, cT):
 foreach $cE : \{dE, cT\}$ **do**
 if dE has children **then**
 foreach $cE_i : \{cE, cT\}$ **do**
 $dE_i : \{cE_i\}$;
 $cE.cV += \text{computeAggregatedCost}(dE_i, cT)$;
 end
 return $cE.cV$;
 else
 return $cE.cV$;
 end
 end
end Function

Algorithm 2 Computing the Values of Aggregated Cost Entities.

Data: design Entity (dE), cost Entity (cE), estimation Relationship (eR), cost Value (cV)
Function computeCompositeCost (dE):
 $cE : \{dE\}$
 foreach $eR_i : \{dE\}$ **do**
 if $cE_i : \{eR_i\} \neq cE$ **then**
 $cE.cV += cE_i.cV$;
 end
 end
 return $cE.cV$;
End Function

overall cost value of the system (root composite cost entity) or its subsystems (intermediate composite cost entities). Specifically, the value of a composite cost entity is computed by automatically summing up the values of all its same-level children entities.

Fig. 2 depicts the design entity hierarchy (left side) as well as the corresponding cost entity tree (right side). The aggregated cost entities whose values are automatically computed via hierarchical summation are blue-colored, the composite cost entities are green-colored, and individual cost entities, whose values are computed by designer-applied functions, are pink-colored. The white elements represent the design entities of a system.

In Fig. 2 we present the hierarchy of cost entities; this hierarchy pertains also to the order of computations. In order to show how aggregated and composite cost entities are computed, we employ the following functions (in pseudocode format).

Function 1 is used to compute the value (cV) of an aggregated cost entity (cE) –of a certain type (cT)– corresponding to a specific design entity (dE). It thus receives as input the design entity and cost type, and generates as output the corresponding value of the aggregated cost entity. It is worth mentioning that we exploit the custom relationship (i.e., estimation) that is uniquely used to connect design entities to cost entities.

Specifically, based on the estimation relationship, a cost entity that has the input cost type and is connected to the input design entity is first found and set as our current cost entity. In case the associated design entity has no children design entities (cDE_i), i.e., it is not composed of others, the function terminates and the cost value of the current cost entity is returned. In case the design entity has children, we apply the function recursively on each of the cost entities (cCE_i) which estimate them, as long as they are of the same –input– type; the cost value is updated via summing up the return values of the recursive functions calls.

Function 2 is used to calculate the value (cV) of a composite cost entity (cE) corresponding to a specific design entity (dE), which is the only input of the function. The generated output is the cost value of the corresponding cost entity. We employ the estimation relationship (eR_i) that the design entity has with cost entities, in order to locate our current cost entity (cE_i). By traversing all the rest of the estimation relationships (i.e., between the design entity and the corresponding aggregated or individual cost entities), we extract their cost value and sum them up to compute the value of the composite cost entity.

E. Cost Computation Functions Inventory

The inventory of the cost computation functions is essentially a collection of various mathematical or logical expressions that transform input (e.g., structural property values of a system component) to output (i.e., the cost value of this component). Extracting the values of individual cost entities is equivalent to computing these expressions.

Note that multiple functions may be readily available within the inventory, loaded to our custom cost profile (see Section III-B). Forming such an inventory brings the following benefits to the system designer. First, functions are readily available in a drag-and-drop form for connection with appropriate cost entities. This enables function reuse without having to reimplement them from scratch. Even if the designer is not a cost-analysis expert, he/she can utilize them to perform a useful cost analysis. Second, the inventory can be easily extended with more functions at will (by the designer himself/herself or another expert), e.g., to perform more fine-grained cost analysis. Third, it can also be used across several system domains, containing computation functions for various cost categories. Last but not least, during system design, the inventory remains a part of the modeling environment, meaning that all computations are defined and conducted within a single tool, without requiring time-consuming interactions with external cost-computation tools.

IV. STUDYING SYSTEMS FROM A COST PERSPECTIVE

To explore the applicability of the proposed approach and the efficiency of the SysML cost profile, two case studies were employed using real-world systems: 1) an REMS from the healthcare domain, and 2) the Athens Metro from the public transportation domain. Each case study deals with a specific cost-related issue in a different system, modeled using SysML. Both case studies are described in the following structure:

- 1) The purpose of each case study and the problem to be solved via the proposed approach.
- 2) The SysML model of each system under study is described. It consists of the structural elements representing the system and the imposed user requirements.
- 3) The way the SysML Cost profile is applied in each system to perform cost analysis is explained. Sample cost functions used to compute individual cost entities are also presented.
- 4) Results extracted from the case study design problem analysis are presented and discussed.

A. REMS Case Study

REMS [46] enables the remote monitoring and diagnosis of the medical condition of elderly individuals from professional health caregivers (e.g., doctors) [47]. Patients agree to install specific devices in their homes, connected to a remote health monitoring facility (e.g., a hospital), while they may also wear devices to monitor their health, thus, become part of the system itself [48], [49], [50].

Our experience stems from the heart monitoring service developed as part of the EMBIoT research project and operated at a prototype level by the Hamad Medical Corporation in Qatar [51]. In this case, after the patient–doctor consultation, a REMS solution designer working in Hamad helps patients choose from a list of tested and approved solutions to be installed in their homes, taking into consideration the necessary properties identified by the doctor (minimum requirements) and patient concerns [52]. To enhance the patients' willingness to become part of the system, their concerns should be taken into account and transformed into design requirements that the system should satisfy [18].

Such concerns may be considered as high-level requirements, targeting comfort or affordability, while at the same time the performance of the configuration should serve their conditions. Consulting with the designer, patients may prioritize their concerns and make decisions on the suggested configuration. The process is presented in [18]. In the following, we focus on affordability concerns, e.g., the cost of the devices patients should purchase and install in their home. We explore how the SysML cost profile may contribute in the computation of the cost of specific configurations and, consequently, enable patients to prioritize affordability concerns with the assistance of the system designer.

In the context of EMBIoT project, interviews were done with face-to-face sessions, although other alternatives may also be adopted. The exact interaction time depends on the use case and is not predetermined, while the interactions are coordinated by the designer. We realized the medical staff were not willing (and equipped) to consider technical details of the devices installed in patients' homes, thus the involvement of REMS engineers was helpful.

1) *Problem Statement:* As part of EMBIoT research project [51], volunteers after a cardio-surgery may recover at their residence, while remotely monitoring their physiological signs. However, each patient's monitoring needs may vary, resulting into different configurations of REMS corresponding to different acquisition budgets [18].

A configuration is considered suitable only if it fits patient health monitoring requirements, which reflect upon primary patient concerns. However, patients need to reflect on their concerns (expressed as criticalities) and realize the fact that it might not be possible to satisfy all of them. Enabling them to understand conflicts between concerns, including affordability, and consciously participate in their prioritization, reflects on their willingness to use such a system [52].

In order to do that, the REMS solution designer creates an REMS Home SysML model, as described in detail in the

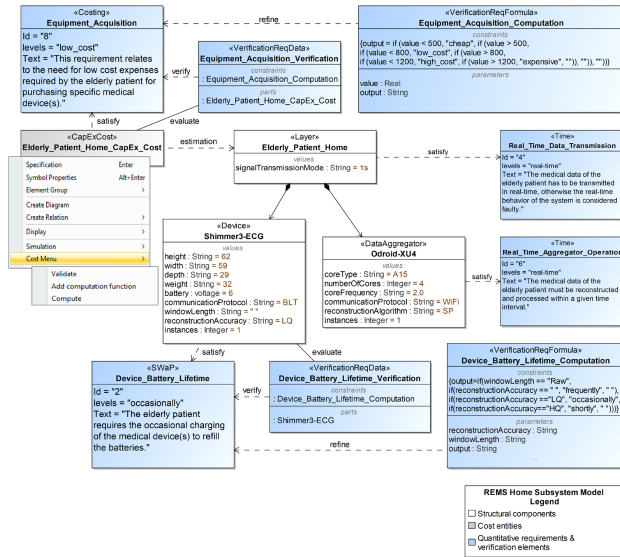


Fig. 3. REMS Home model.

following section. The SysML cost profile enables a personalized cost analysis; it provides the REMS designer with the necessary tools to assess costs at different levels, i.e., from the overall REMS CapEx cost to the individual acquisition expenses of medical and data communication devices, in an automated fashion.

2) *REMS Home SysML Model*: Two basic components are defined in the REMS Home model (see Fig. 3): 1) Shimmer3-ECG [53], i.e., the medical device, comprising an electrocardiogram (ECG) sensor; size, battery, communication protocol, or signal reconstruction accuracy are some of the properties of the sensor device that characterize it. 2) Odroid-XU4 [54], i.e., the data aggregator; properties such as type of CPU core or number of these CPU cores assist the designer to fully describe this entity.

To depict patient concerns, the designer defines requirements attached to REMS components. They are described by text and a quantitative levels property, whose value indicates the desired level that covers patient needs [55]. This way, it is easier for patients to describe their design requirements. However, this information is too abstract to result in analytical system configurations. Thus, each requirement is refined by the designer to describe requirement levels, by formulas which are computed based on properties of the REMS components. These formulas are described by the designer and are applied to different patient configurations. They are stored in SysML VerificationReqFormula entities, refining requirements, as shown in Fig. 3. VerificationReqData entities are defined to store the actual computed value of each requirement's level, corresponding to a specific configuration. Their comparison to desired levels verifies requirement satisfaction [18].

Indicatively, the Device_Battery_Lifetime requirement is associated with the Shimmer sensor in Fig. 3, holding the levels property with an “occasionally” value, meaning that occasional charging is required to refill the battery of the Shimmer device. The VerificationReqFormula entity refining

this requirement provides the formula for computing the Device_Battery_Lifetime requirement level.

The Equipment_Acquisition is a cost-related patient requirement (it is termed costing, as shown in Fig. 3), indicating the level of expenses required by an elderly patient for purchasing an REMS Home configuration. Thus, it should be associated to the Elderly_Patient_Home entity. However, this is a cost requirement. Thus, since there is a need to assess the costs of the overall solution, the designer creates a composite CapExCost entity (see Elderly_Patient_Home_CapEx_Cost in Fig. 3) that represents the total capital expenses of the REMS Home, and connects it (via estimation relationship) to the top component of the structural hierarchy, i.e., the Elderly_Patient_Home. Note that this is a composite CapExCost entity, which shall be computed in a latter stage. It is created at this stage to associate the Equipment_Acquisition requirement with it.

Note that in our modeling tool, i.e., Cameo Systems Modeler (CSM) [56], the CapExCost entity is accompanied by a Cost Menu that contains specific buttons; these buttons correspond to steps that the designer can follow in order to perform a complete and accurate cost analysis. Specifically, the designer can 1) automatically generate cost entities for the child structural components, 2) add computation functions to extract the values of individual cost entities, and 3) automatically compute the values of aggregated and composite cost entities. These steps are described in detail in the following sections.

The Equipment_Acquisition requirement, depicted in Fig. 3 is of a “low_cost” level. Thus, based on the formula, defined in the Equipment_Acquisition_Computation VerificationReqFormula, the acquisition cost of the equipment for the patient should not exceed “800” Euro. Is this possible taking into account the other requirements set for the patient? The operation of the system should be performed in “real time” to meet the patient’s healthcare needs, while the patient wished to have to charge the sensor “occasionally.” How are cost requirements related to other requirements? All of them are estimated based on the configuration of the system, e.g., the properties of the Shimmer3-ECG and Odroid-XU4 components. Our SysML Cost profile should assist the designer to associate CapExCost cost entities to these components, specify functions for the cost estimation of each of them, and automatically compute the values of Elderly_Patient_Home_CapEx_Cost already defined by the designer. Note that this can be based on automated–recommended– pre-configured setups, as presented in [57].

3) Performing REMS Home Cost Analysis:

a) *Generate Cost Entities*: The CapExCost entity, connected to the Elderly_Patient_Home layer, is composed of aggregated costs, i.e., the AcquisitionCost and IntegrationCost entities, connected to the same component. In addition, each child of the layer, i.e., the device and the data aggregator, should be connected to respective AcquisitionCost and IntegrationCost entities. Note that the AcquisitionCost is chosen to demonstrate the cost of purchasing these components, while the IntegrationCost represents the cost of integrating them, according to the SysML Cost profile (see Fig. 1). The profile facilitates the automatic generation of subcost entities from a composite one, in this case the Elderly_Patient_Home_CapEx_Cost (see Fig. 4).

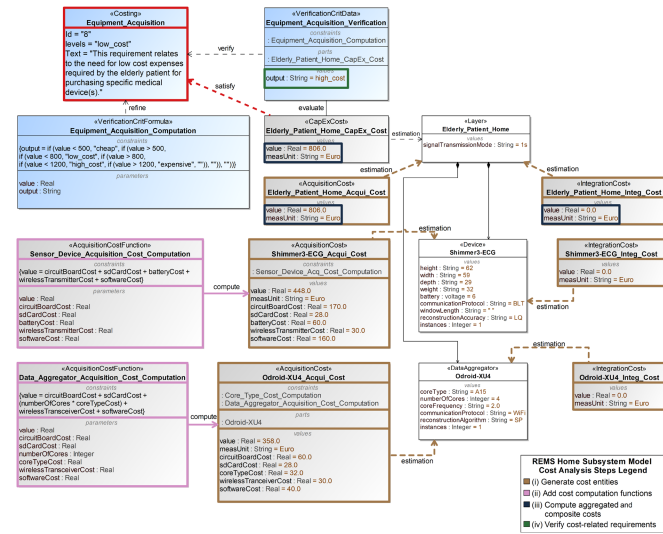


Fig. 4. CapEx cost computation in the REMS Home model.

They are automatically created, initialized, and connected to system components; even if the designer neglects the insertion of a cost entity, our approach integrates it automatically to the model. In the REMS Home case, due to the minimal number of components, the contribution of the IntegrationCost entity to the overall CapEx is negligible. Although cost entities are defined automatically, the system designer should add their properties used for their computation, as explained in the next section.

It is worth mentioning that all REMS-related data were provided by the Hamad Medical Corporation and the College of Engineering at Doha University, in Qatar, as part of the EMBIoT research project [51]. Due to the sensitive nature of this medical data, it remains proprietary; we were allowed to disclose only high-level (anonymized) insights and information. In addition, according to the hospital’s policies, devices’ cost data remain constant on a yearly basis. Note that, irrespective of change frequency, the system designer can easily incorporate any data-related alterations (e.g., the price of a REMS device may be increased or decreased) within the model. Computations encapsulating supply-chain backlogs, back orders, etc., or even promotional discounts, can be inserted in the model in the form of additional functions by the system designer according to his/her modeling objectives.

b) Add Cost Computation Functions: The cost entities connected to the device and the data aggregator are considered as individual, i.e., their cost values are computed one by one by custom cost computation functions; in our case, only the AcquisitionCost entities are computed, while the values of the integration costs are set to zero.

When our SysML Cost profile is applied to the REMS home system model, functions from the inventory are readily available for computation; these functions are solved within the modeling tool in SysML parametric diagrams, contained in respective individual cost entities, using OpenModelica [58] as the (math) solver. Indicatively, the parametric diagram for Odroid-XU4 acquisition cost is illustrated in Fig. 5. Note that input properties from the cost entity and system components may be used for the computation of the function, while the output is the resulting

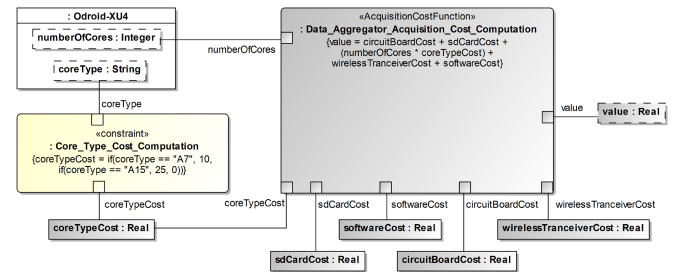


Fig. 5. Odroid-XU4 data aggregator acquisition cost computation.

cost value, stored within the cost entity. It is worth mentioning that our approach supports dependencies between functions; for example, in Fig. 5, there is a secondary function within the same parametric diagram (see yellow-colored block) which produces the value of a property that is in turn used as input to the main cost computation function. Here, the cost of different core types is first computed and then used to compute the cost value of the multicore Odroid data aggregator.

c) Compute Aggregated and Composite Costs: Following the addition of the required cost computation functions, the latter are automatically computed, while respective cost values are extracted and stored to associated cost entities. In turn, based on these values, the values of aggregated costs can be computed via simple summations. The AcquisitionCost entity, connected to the Home layer, is aggregated; therefore, it obtains its value only from the summation of the Acquisition-type costs of the device and the data aggregator. The same applies to the respective integration costs. The CapExCost entity of Home layer is derived by the summation of the values of the aggregated cost entities.

d) Verify Cost Requirements: In order to verify the Equipment_Acquisition costing requirement depicted in Fig. 4, the value of the CapExCost entity is employed; the corresponding VerificationReqFormula entity receives this value as input, initiating the verification process, while the actual value of the cost-related level is computed within it (via parametric execution). In the case of REMS home model presented in Fig. 4, CapExCost value is “806” Euro, resulting in a “high_cost” level of the Equipment_Acquisition costing requirement. Thus, the requirement may not be met, thus, it is automatically framed with red color.

Note that when there is a verification problem, our approach enables the modeling environment to automatically exhibit appropriate information to the designer, as well as adjustments that he/she can make. The latter may include suggested actions or solutions (like reconfiguring the system) to reduce costs. This process can occur several times –dynamically– in the design process of the system, adjusting to changing operational conditions [18].

4) Results and Discussion: Patients need to reflect on their concerns and realize the fact that it might not be possible to satisfy all of them. The designer transforms high level descriptions corresponding to patients’ concerns, into design requirements and properties as depicted in Fig. 4. After verifying these requirements against possible system configurations, the designer discusses with the patient the satisfaction of the concerns and, when in conflict, helps them explore their prioritization. In the

TABLE I
EVALUATING THE TRADEOFF BETWEEN COST AND PERFORMANCE LEVELS IN
THE REMS HOME SYSTEM

CapEx cost (€)	Cost limit (€)	Afford. level	QoS level			Accepted solution
			Mode of operation		Battery lifetime	
			Device	Data aggregator		
631	500-800	low-cost	best effort	best effort	occasionally	X
631	500-800	low-cost	best effort	best effort	frequently	X
756	500-800	low-cost	best effort	real-time	frequently	X
806	800-1200	high-cost	real-time	real-time	shortly	X

case of Fig. 3 there are three patient concerns explored: 1) the mode of operation, transformed to performance parameters (how often are patient data collected), 2) comfort (e.g., battery life time), and 3) affordability (related to the cost of the solution).

Table I contains alternatives for the levels of these concerns corresponding to REMS configuration presented in the same figure. The real-time operation and battery lifetime concerns are considered as performance parameters according to Wymore [2]. Thus, the tradeoff between performance and cost for the configuration of Fig. 3 is depicted in the table. All of them provide valid system, e.g., appropriate output according to Wymore.

However, the system designer is interested in the acceptable ones, that satisfy both performance and cost concerns of the patient. In this case, patient's mode of operation and battery lifetime concerns, and affordability concerns, e.g., the cost restriction is the patient's budget for the acquisition of REMS devices. The acceptability of a solution on both fronts is explicitly stated in the last column of the table (i.e., "Accepted solution" column).

In this case, the patient asked for a "low-cost" solution, while appliances should operate in "real-time" mode and charge batteries "occasionally." As shown in Table I, there is no acceptable solution, e.g., design output under the technology employed in the EMBIoT project, that may satisfy all three concerns (all cells of the specific row should be transparent). Compatible alternative matching for them is depicted in the table, though nonsatisfied patient concerns are set in Fig. 3. The designer may share this data with the patient, explore the conflicts between their concerns and help them prioritize them. Real-time operation is power consuming and cannot be achieved without charging batteries in a "short" period of time. Furthermore, the primary CapEx driver of the REMS is the real-time operation requirement, since the battery behavior does not have any impact on cost (see first two rows of the table). In case the elderly patient requires a "low-cost" CapEx level, the patient should be willing to compromise on the real-time operation of the medical units (which should measure, process, and transmit medical data as quickly as feasible). If "real-time" operation is a prerequisite for the patient's condition, cost is high and batteries should be charged shortly. To meet "low-cost" and "occasionally" battery charging, the patient's condition should allow "best-effort" data transmission and monitoring. This is true in most of the cases in the EMBIoT case study. Summarizing cost versus performance exploration data in a format the patients understand, enables them to make more conscious decisions on how to use REMS, always with the assistance of the designer.

On the designers' side, the proposed approach enables them to automatically compute CapEx of specific configurations, and

categorize it into specific affordability levels (see the first, second, and third row of Table I) integrated into the patient concerns discussion.

In the specific example of REMS case study, performance and cost constraints set by the patient were not satisfied. However, the application of the proposed approach provided the necessary data to enable patients explore the tradeoff between conflicting QoS and affordability concerns. This case is indicative of the fact that finding the right balance between cost and performance is not trivial, even in a simple REMS scenario with two devices. Patients may be inclined to pay more to address their primary concern, such as real-time operation. However, they have to decide whether they really need it or not, based on their condition and their doctor's advice.

B. Athens Metro Case Study

The Athens Metro, operated by the ATTIKO METRO, S.A. [59] and Urban Rail Transport, S.A. [60], is an RTS that provides public transportation services in the metropolitan area of the capital city of Athens in Greece, populated with approximately 5 million people.

In the following sections, we focus on addressing a fundamental challenge in large-scale RTSs, which is the need for adapting operations to satisfy commuters (e.g., in terms of available space within trains and stations), while minimizing the additional operating costs that this adaptation involves. We select the Athens Metro as our case study.

The operators of the Athens Metro aim to provide efficient transportation services to the commuters (i.e., the end-users of the RTS), in multiple dimensions. The services provided by an RTS are rated by international standardized performance parameters, defined as LoS. Each of these parameters is rated from "F" to "A," based on the formulas provided by [61]. One of the most popular LoS parameters is passenger comfort, e.g., the spatial comfort of passengers while waiting at a stop or within a moving train.

In collaboration with Athens Metro operators we estimated the passenger comfort for all the lines of Athens Metro system and explored ways to improve it; for more details we refer the reader to the work of [20] where comfort is quantified in terms of LoS and is measured under varying traffic conditions. Our study enabled the operators to estimate passenger comfort during peak hours, i.e., 8–11 A.M. (business times) for all the lines and stations. We also explored whether there was a possibility to improve passenger comfort, without any modifications in the existing infrastructure. During the peak hours, Athens Metro train routes take place every 7 min, yielding a LoS "D." This is a barely acceptable LoS (assuming "A" as the best and "F" as the worst). The operators want to explore the improvement of passenger comfort with the existing infrastructure. Using simulation, we concluded that by changing only the frequency of the train routes, a "B" could be obtained. This prospect has no additional acquisition cost, but what about the operational cost of the Metro system?

To explore this problem, we applied the SysML cost profile in the existing SysML model which is used to simulate and study Athens Metro System [20], as described in the following:

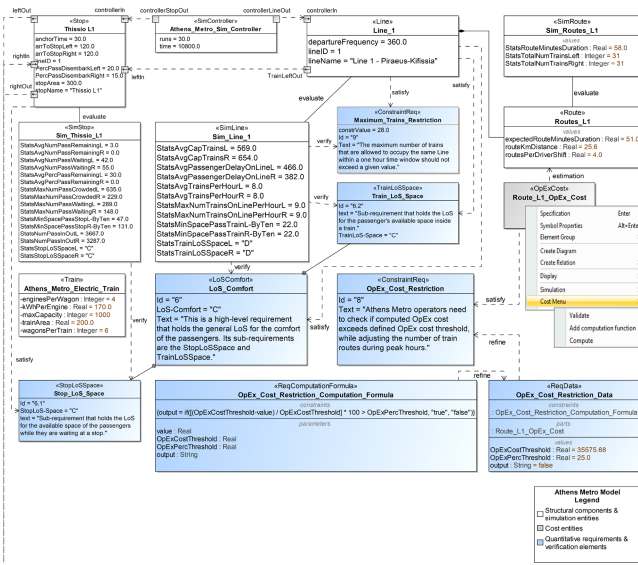


Fig. 6. Athens Metro SysML model.

1) **Problem Statement:** Altering the operational conditions of the system to achieve better LoS, e.g., increasing the frequency of train routes during peak business hours from 7 up to 3 min, comes at an extra –operational– cost. The increase in operational costs is associated with the additional number of train routes which involve more drivers (and thus, paid shifts), more energy consumption and more maintenance costs. Therefore, operators require the evaluation of the benefits in comfort versus the –relative– cost increase, needed to yield these benefits. The Athens Metro operates under specific budgetary constraints; therefore, cost drives company-wide decisions. This is reflected with a specific threshold for acceptable OpEx beyond which increasing LoS is simply not viable.

Thus, altering route frequencies requires two things: 1) a LoS analysis to ensure that the change achieves the required passenger comfort level (as described in [20]), and 2) a comprehensive cost analysis, in terms of incurred OpEx.

Having identified the two facets of the problem, an Athens Metro engineer comes into play. Using the SysML RTS profile, developed in [20], he/she employs SysML within the Cameo modeling environment, and runs simulations to compute comfort LoS under different conditions. This corresponds to Athens Metro exploitation focusing on performance parameters and technology (e.g., infrastructure) constraints according to Wymore [2]. With this at hand, cost parameters should also come into the equation. The engineer employs the SysML cost profile and cost computation mechanisms, which facilitate forecasting LoS and OpEx incurred during peak hours for different train route frequencies; this deals with the second facet, and is the primary focus of the following sections.

2) **Athens Metro SysML Model:** In Fig. 6, we present an excerpt of the Athens Metro system model where the RTS profile, described in [20], was applied and LoS analysis was conducted. In the figure, an abstract view of Line_1 of the Metro system is presented.

The depicted transparent (white) elements represent structural entities of the Athens Metro system; for demonstration purposes we consider one entity per type (e.g., line, stop). Specifically, Line_1 is a primary entity of this system, that contains the routes along which Athens Metro electric trains move. Since LoS needs to be evaluated on a line-by-line basis, the LoS requirement, referring to the general level of passenger comfort inside a train or at a stop, is connected to the Line_1 entity. Line_1 also comprises a sequence of stops. Thissio_L1 is such a stop, corresponding to the platforms of the Athens Metro system and representing the area where passengers are accumulated in order to board an incoming train, or disembark. The Thissio_L1 stop is connected to an LoS-comfort subrequirement, holding the comfort level of passengers waiting at the platforms of the stop. The Athens_Metro_Electric_Train entity represents the trains that carry passengers, providing the transportation service(s). Specifically, after the passengers board a train (e.g., at the Thissio_L1 stop), the train transports them to the next adjacent stop, following a specific route (dictated by Line_1). The Routes_L1 entity represents the course of the Athens Metro electric trains, traversing Line_1, from its starting stop to the final one. It is worth mentioning that routes “glue” together lines, trains, and stops and are important both for the LoS and cost analysis.

The model also includes simulation-related entities, storing simulation results. They are designated as Sim entities. The estimated passenger comfort LoS is stored as one of their properties, after the execution of the simulation. The Sim_Line_1 evaluates the defined Line_1, holding statistics about the number of the moving electric trains, their average capacity, as well as the minimum available space of the passengers inside the trains and the average comfort LoS. The Sim_Thissio holds indicators related to average or maximum number of commuting passengers, as well as the minimum available space around them while waiting at the stop platforms. The Sim_Routes_L1 simulation-related entity evaluates the Routes_L1, holding the statistical values of the performed train routes (e.g., the total number of routes, performed on the left direction or the exact duration of the routes in minutes). This entity’s statistics are informative for the cost analysis that will follow; e.g., the simulated number of routes is factored in while computing the total OpEx. The Sim Controller is used to control the system’s simulation, in particular its time window (180 min, representing the 8–11 peak hours) and repetitions.

In the case presented in Fig. 6, trains pass in Line_1 every “6” min (“360” s), while passenger comfort inside trains is rated as “C.” The same is true for station Thissio_L1, however, the aggregated passenger comfort for Line_1 is rated as “D,” as most stations’ passenger comfort is rated as such. Quantitative requirements regarding comfort were defined, holding a property that described the desired levels (depicted as blue-colored in Fig. 6). The Los_Comfort, depicted in Fig. 6, for Line_1 is of level “C.” It is associated to both the Line_1 entity and all stations and trains that belong to it.

To also integrate cost into Line_1 model, an OpexCost entity should be associated to a structural entity. Routes_L1 is the primary contributor of the Athens Metro OpEx; electric trains that

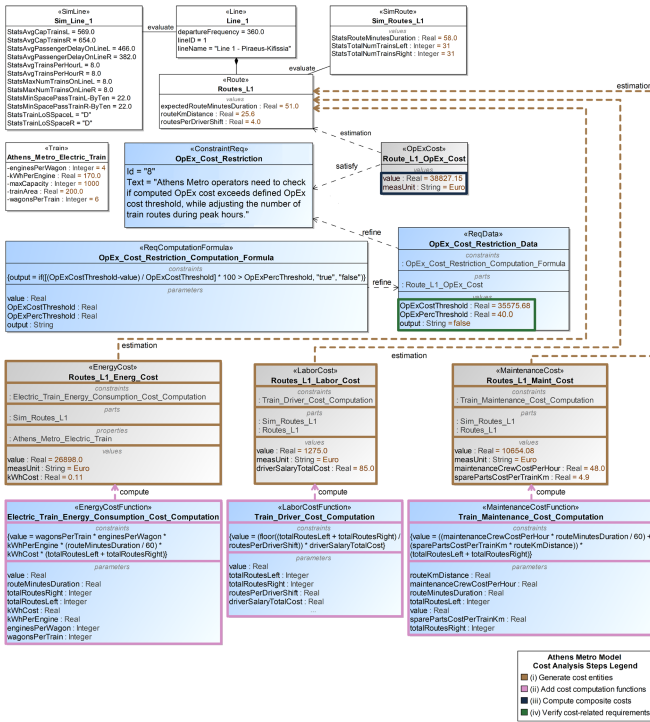


Fig. 7. OpEx cost computation in the Athens Metro system model.

traverse Line_1 routes incur energy consumption, maintenance, and driver crew costs, which compose the biggest part of the OpEx of the Athens Metro, induced during peak hours. Note that each system component holds its own structural properties; some of them may be descriptive (e.g., the ID of a line or the name of a stop), while others, e.g., route properties, may be critical for conducting a cost assessment of the current system operation (e.g., the kilowatt-hours that an electric train consumes or the distance and expected duration of a route).

In Fig. 6, the Routes_L1 entity is estimated by the Routes_L1_OpEx_Cost composite cost entity that represents the overall OpEx of the routes. Note that within our modeling environment the OpExCost entity is accompanied by a Cost Menu; this menu contains the same buttons as in the REMS case study, enabling the engineer to follow specific steps to automatically generate cost entities, add computation functions, and automatically compute costs. These steps are described in the following Sections.

The OpEx_Cost_Restriction requirement is defined, representing the need of the system operators to keep in check the relative differences in operational costs while adjusting the number of train routes during the Athens Metro peak hours. This is important since, assuming a congestion event at a platform (which degrades LoS), system operators may decide to increase the number of serving trains (and thus, routes) to bring LoS back to acceptable levels. However, more routes mean more OpEx (e.g., energy consumption, drivers, etc.). In order to check whether this requirement can be satisfied or not, this OpEx cost is compared against a metro-operator-defined threshold. The relative difference in OpEx costs, upon adjusting the route frequencies, can be defined as follows: $\frac{OpExCost(routes_final) - OpExCost(routes_init)}{OpExCost(routes_init)}$

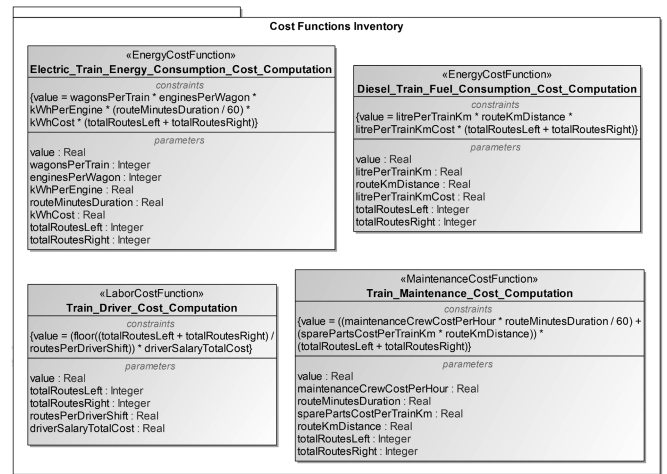


Fig. 8. Cost functions inventory.

Thus, the requirement translates into the following inequality check: $OpEx_Cost_Rel_Diff(routes_init, routes_final) > OpExThreshold$, where the threshold for the relative cost differences is provided by the Athens Metro operation company. This formula is depicted in Fig. 6 within the OpEx_Cost_Restriction_Computation_Formula, which results in a “true”/“false” output value, indicating that the final operational cost value exceeds (or not) the defined threshold. The OpEx_Cost_Restriction_Data is used to refine the costing requirement; it is the holder of two values provided by the system operators, i.e., the existing (or initial) OpEx cost and the desired threshold. It also contains the computed result of the computation formula.

3) Performing Operational Cost Analysis:

a) Generate Cost Entities: The OpExCost entity, defined in the Athens Metro model (see Fig. 7), is composed of the following individual cost entities: 1) the Routes_L1_Energy_Cost, which refers to the energy consumption cost incurred by the electric motors of the Athens Metro electric trains moving along the Line_1 route, 2) the Routes_L1_Driver_Cost that is associated to the wage of the operating driver(s) (working in shifts), and 3) the Routes_L1_Maintenance_Cost, which refers to the expenses incurred by maintaining (e.g., repairing if needed) the traversing Line_1 electric trains. Note that each cost entity holds cost-specific properties, critical for their cost value computation (e.g., the wages of a train driver is reflected in the driverSalaryTotalCost property of the Routes_L1_Driver_Cost entity).

b) Add Cost Computation Functions: Following the definition of the cost entities, the engineer can use functions, that are readily available in the cost inventory (see Fig. 8), to compute energy consumption, labor, and maintenance costs for the electrically powered trains that move along the Athens Metro routes. In Fig. 7, we illustrate these functions within the Athens Metro model.

The energy cost value of the electric trains traversing the route is the result of the multiplication of the 1) number of wagons per electric train, 2) the number of engines per wagon,

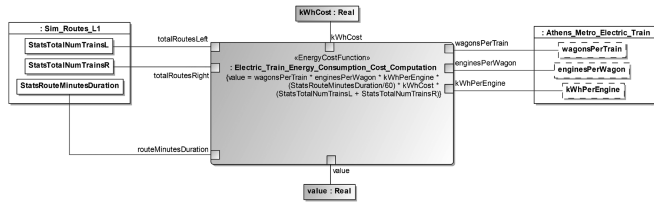


Fig. 9. Electric trains energy consumption computation.

3) the kilowatt-hours that electric train engines consume (per hour), 4) the duration (in hours) of a complete route, 5) the cost of each consumed kilowatt-hour, and 6) the total number of performed routes (the summation of the number of routes, performed in both directions). Regarding the cost value of a train driver, it is the result of the multiplication of 1) the total number of routes divided by the number of routes that a driver performs on his/her shift, and 2) the total salary cost of the driver. Finally, the maintenance cost value for electric trains is the result of the summation of 1) the maintenance crew cost per hour, multiplied with 2) the duration of a complete route (in hours), and 3) electric train spare parts costs per train-Km, multiplied with 4) the route's distance (in km); this summation is multiplied with 5) the total number of performed routes. Note that these functions are contained in the cost profile functions inventory depicted in Fig. 8.

In order to compute each cost value, parametric diagrams provided by SysML, are employed; Fig. 9 depicts such a parametric diagram holding the equation for the computation of the energy cost value of an electric train. The equation takes input properties whose values are used to compute it; these properties may belong to different entities. In particular, the energy consumption cost function takes input from 1) the Athens_Metro_Electric_Train entity, 2) the Sim_Routes_L1 entity, and 3) the Routes_L1_Energy_Cost entity. The Athens_Metro_Electric_Train provides the numbers of the train wagons, their electrical engines, and the energy consumption of these engines in kilowatt-hours. The Sim_Routes_L1 entity supplies the duration of a complete route and the total numbers of routes, performed in both directions, during peak hours. Finally, the Routes_L1_Energy_Cost entity provides the cost of a kilowatt-hour; note that this is a cost-specific input property. The equation is solved within the parametric diagram (using OpenModelica as the math solver) and the output energy cost value is extracted for further cost assessment. Here, the value property represents the extracted output value of this equation. Using a parametric diagram within the modeling environment, the engineer does not have to go through the complexity of writing additional code for the computations. He/she simply provides the corresponding input values and defines the functions that will be used; after the computations are complete, the final required cost values are easily extracted and may be used for the cost analysis of the system.

c) Compute Composite Costs: Since the aforementioned functions refine a respective cost entity, the computed output value is stored within these cost entities. According to our approach (see Section III-D), an OpExCost entity obtains its

TABLE II
EVALUATING THE TRADE-OFF BETWEEN COST AND PERFORMANCE IN RTS

Athens Metro Line 1	Routes (L + R)	Peak hours OpEx (E)	Peak hours OpEx increase (%)	Daily OpEx (E)	Daily OpEx increase (%)	Avg LoS	Accepted solution
Line 1 trains frequency (minutes)	7	56	35575.68	—	160494.03	—	D
	6	62	38827.15	9.1	164213.1	9.14	D
	5	78	49551.84	39.2	174243.98	15.8	C
	3	128	81315.8	128.5	205632.99	36.7	B

value from the sum of the values of EnergyCost, LaborCost, and MaintenanceCost entities via automatic summations. Therefore, at this stage of the analysis, the engineer has all the components and the method needed to automatically compute the OpEx-Cost value. Indicatively, in Fig. 7, the Routes_L1_Opex_Cost obtains a “38827.15” Euro cost value (assuming a “6”-min train interarrival interval); this is the result of the summation of the Routes_L1_Energy_Cost, Routes_L1_Driver_Cost, and Routes_L1_Maintenance_Cost entities' values during the defined peak hours, when the Athens Metro was studied.

d) Verify Cost Requirements: In Fig. 6, the desired LoS for Line 1 was set at a “C” level, assuming a “6”-min route frequency). The simulation returned an LoS “D.” Thus, LoS_Comfort requirement (see Section IV-B2) was not satisfied for Line_1 entity. What about operational cost restrictions, set to “40”% increase limit? The requirement is verified in the same fashion, as depicted in Fig. 7. As indicated in OpEx_Cost_Restriction_Data refining the cost requirement, the initial operation cost is “35575.68” euro for Line_1 in peak hours between 8 and 11 each morning, while the acceptable operational cost increase is up to “40”%. Thus, the solution is acceptable costwise, although the desired Passenger comfort LoS in not achieved. Related entities are not highlighted, since the cost requirement is satisfied.

4) Results and Discussion: Table II contains alternatives for passenger comfort LoS and operational cost increase corresponding to different train frequencies for Line_1. The engineer runs simulations to calculate LoS for different train frequencies (corresponding to “7,” “6,” “5,” “3”-min intervals between arriving trains) during rush hours. All of them provide valid system (i.e., appropriate) outputs according to Wymore [2]. However, Athens Metro engineer is interested only in the acceptable ones, that satisfy both performance, e.g., passenger comfort LoS, and cost restrictions. That is, passenger comfort LoS should be set to “C” for Line_1 with an increase of up to 40% for peak hours to the operational cost (see Fig. 6).

Having yielded suitable solutions for the Metro peak business hours (8-11 A.M.), operational cost data may be extrapolated on a daily level. Metro operators have a dual OpEx requirement, concerning the peak hours (max 40% relative OpEx increase) and the daily operation (max 25% relative OpEx increase). Thus, daily OpEx increase is also depicted in the table.

In essence, this table contains alternative system design solutions achieving certain operational expenditures and LoS. In the context of RTS in particular, the system design goal is to improve LoS for passengers, and the cost restriction is a maximum relative OpEx increase over which this improvement is not financially viable. The acceptability of a solution on both fronts is explicitly stated in the last column of the Table (i.e.,

TABLE III
CASE STUDIES CHARACTERISTICS

Case study	REMS configuration	Athens Metro operation
Case study purpose	Integrate patient concerns into REMS solution design and help them prioritize them	Improve LoS during rush hours taking into account cost restrictions
Cost management approach target	Compute CapEx of specific solutions to explore affordability concern	Compute OpEx under different operation scenarios
Stakeholders involved	Patient, REMS solution designer	Operator Engineer
Contribution	Assist patients to explore tradeoffs between affordability and quality of service concerns	Forecast performance and cost of specific scenarios
Difficulties encountered	Transform abstract concerns into computable requirements	Validate operation cost computation functions

“Accepted solution” column). We observe that configurations from and under “5” min train frequency achieve better LoS (“C” or “B”), however the “5”-min configuration is the optimal solution balancing LoS and cost constraints (both for peak and daily hours). Time intervals under “5” min (i.e., “3” min) lead LoS to an even better level (i.e., “B”), however, the relative OpEx cost increase it demands violates the constraints set for both the peak hours and the daily operation.

In this case, our approach assisted the Metro Operator engineer to easily forecast LoS and OpEx and identify an acceptable solution with the proper balance between performance and cost. This assessment takes place in a single graphical modeling environment, where both performance and cost estimations are integrated within the SysML model and presented to the user. Quantifying the tradeoffs is key.

It is certainly not a trivial task to increase patient comfort in a large-scale RTS. It costs money and it is often not a first-level priority for the operators. In the case of Athens metro, we provided alternatives for LoS configurations and corresponding operational cost. Revenue and profit exploration was not part of the study discussed.

V. DISCUSSION

Here, we discuss the contributions of the proposed approach applied in two distinct case studies with different purpose and characteristics.

The purpose of the Athens Metro case study is to improve LoS during rush hours taking into account cost restrictions. Our contribution in this case is forecasting performance and cost during rush hours, e.g., exploring how to improve LoS with an acceptable cost increase. Integrating cost management into SysML enables the computation of both performance and cost of a specific scenario at the same time. An Athens Metro team of engineers was responsible for exploring different scenarios.

The purpose of REMS case study is rather different, i.e., to integrate and prioritize patient concerns into system design. Patients need to reflect on their concerns (expressed as criticalities) and realize the fact that it might not be possible to satisfy all of them. An REMS solution designer transforms high level descriptions corresponding to concerns into design requirements and explores the cost of corresponding solutions. After verifying these requirements against possible system configurations, the designer discusses with the patient the satisfaction of the concerns and, when in conflict, helps them explore their prioritization. Computing CapEx of a selected configuration is rather straightforward in this case, while no forecasting is needed. Nevertheless, our approach enabled the designer to

assist patients to explore tradeoffs between affordability and quality of service concerns. The different characteristics of the two case studies are summarized in Table III.

Calculating cost reflects on the decisions made by different stakeholders when designing or configuring a system, either complex or simple. Regarding the Athens Metro case study, one of the difficulties faced by the engineers was the formulation of the operation cost computation functions and their precision. Comparing the forecasted cost of the existing operation with the actual cost and making proper minor adjustments enabled them to validate the cost computation function. The policy suggested for the rush hours to ensure LoS level C, with trains passing every 3 min in Line 1, was applied by the operator at the beginning of 2021. Obtained data refer to the first half of 2021. The operation cost increased by 41% during rush hours, similar to the forecasted increase. The system LoS level was computed as C, however, the number of passengers using the metro decreased during this period due to COVID-19 restrictions.

Regarding the REMS case study, computing installation costs was rather trivial, since the cost of the approved components was known in advance. However, the main difficulty was to transform the patient’s rather abstract concern, e.g., a affordable solution, into computable quantitative requirements. In EMBIoT project, the experience of talking to patients enabled the REMS solution designer to suggest such a transformation, like the limits of an affordability category, as depicted in Table I. In this case, the patient has to choose between a list of solutions considering the performance characteristics and cost, taking into account their personal medical needs and concerns.

There are limitations identified when applying the proposed approach. Some of them are parts of the method itself. Both the problem under consideration and the system under study must be modeled using SysML. Furthermore, the involved stakeholders, for example the engineer, should have the data and knowledge to produce/depict cost computation functions in different levels of detail. However, aggregations and validations are provided by the method and related profile.

When applying the approach in specific domains or case studies, a thorough knowledge of the domain is required. As shown by the case studies, the efficiency of the approach depends upon the detailed specialization in the specific domain. The engineers must realize what to test against cost constraints. This in practice is not always straightforward, especially when the engineer is nonfamiliar with model-based design. System complexity itself also might be an issue. To this end, we intend to further explore the applicability of the proposed approach in more complex case studies, where more than one design issue has to be explored against cost.

VI. CONCLUSION

Cost, likewise performance, is a crucial system design parameter. By extending SysML to enable cost analysis, this popular language becomes more efficient and comprehensive for MBSD. The proposed approach is general, as the resulting cost profile can be integrated into any system model, existing or built from scratch. Moreover, the automated computation of individual costs (via cost computation functions, included in a reusable and extendable library) assist the designer to perform a cost analysis of SoS, at different levels of granularity, across domains.

We demonstrated the feasibility and the benefits of this approach in two distinct domains with different structural and operational characteristics. In the context of the REMS, patients' concerns were accommodated under specific CapEx restrictions. In the context of Athens Metro RTS, we evaluated LoS improvements and the respective impact on OpEx. In both cases, performance and cost constraints were integrated within the system model, enabling the designer to check them while designing the system.

As part of our future work, we will expand our analysis to more fine-grained cost categories, and provide domain-specific cost computation functions. We further plan to extend the SysML cost profile to support revenue and profit analysis as well. Corresponding properties and functions may be supported and automatically computed in a similar way as cost ones. Furthermore, the integration of external tools, facilitating financial analysis, with SysML modeling frameworks shall also be explored.

REFERENCES

- [1] J. A. Estefan, "Survey of model-based systems engineering (MBSE) methodologies, Rev. B," in *Proc. NCOSE-TD-2007-003-01, INCOSE MBSE Initiative*, 2008.
- [2] A. W. Wymore, *Model-Based Systems Engineering*, vol. 3. Boca Raton, FL, USA: CRC Press, 2018.
- [3] J. Spruytte, M. Van der Wee, S. Verbrugge, and D. Colle, "Modeling equipment hierarchy and costs for ICT solutions," *Trans. Emerg. Telecommun. Technol.*, vol. 30, no. 3, 2019, Art. no. e3583.
- [4] A. Madni and S. Purohit, "Economic analysis of model-based systems engineering," *Systems*, vol. 7, Feb. 2019, Art. no. 12.
- [5] R. Karban et al., "MBSE initiative—SE2 challenge team-cookbook for MBSE with SysML," *SE2 Challenge Team*, 2011.
- [6] M. Dabkowski, J. Estrada, B. Reidy, and R. Valerdi, "Network science enabled cost estimation in support of MBSE," *Procedia Comput. Sci.*, vol. 16, pp. 89–97, 2013.
- [7] D. Gattuso et al., "A tool for railway transport cost evaluation," *Procedia-Social Behav. Sci.*, vol. 111, pp. 549–558, 2014.
- [8] G. Troche, "Activity-based rail freight costing: A model for calculating transport costs in different production systems," Ph.D. dissertation, KTH, Stockholm, Sweden, 2009.
- [9] D. Shermon, *Systems Cost Engineering: Program Affordability Management and Cost Control*. Evanston, IL, USA: Routledge, 2017.
- [10] J. Lane, "Cost model extensions to support systems engineering cost estimation for complex systems and systems of systems," in *Proc. 7th Annu. Conf. Syst. Eng. Res.*, 2009, pp. 1–8.
- [11] Object Management Group, Ed. (2019), *OMG Systems Modeling Language (OMG SysML) Version 1.6 beta*. [Online]. Available: <https://www.omg.org/spec/SysML/>
- [12] S. Friedenthal et al., *A Practical Guide to SysML: The Systems Modeling Language*. San Mateo, CA, USA: Morgan Kaufmann, 2014.
- [13] S. Wolny, A. Mazak, C. Carpella, V. Geist, and M. Wimmer, "Thirteen years of sysml: A systematic mapping study," *Softw. Syst. Model.*, vol. 19, no. 1, pp. 111–169, 2020.
- [14] A. Tsidimas, M. Nikolaidou, and D. Anagnostopoulos, "Extending sysml to explore non-functional requirements: The case of information system design," in *Proc. 27th Annu. ACM Symp. Appl. Comput.*, 2012, pp. 1057–1062.
- [15] "Paramagic plugin," [Online]. Available: <https://www.nomagic.com/product-addons/magicdraw-addons/paramagic-plugin>
- [16] S. K. Sowe, E. Simmon, K. Zetsu, F. de Vault, and I. Bojanova, "Cyber-physical-human systems: Putting people in the loop," *IT Professional*, vol. 18, no. 1, pp. 10–13, 2016.
- [17] M. Jirgl, Z. Bradac, and P. Fiedler, "Human-in-the-loop issue in context of the cyber-physical systems," *IFAC-PapersOnLine*, vol. 51, no. 6, pp. 225–230, 2018.
- [18] M. Nikolaidou et al., "Incorporating patient concerns into design requirements for IOMT-based systems: The fall detection case study," *Health Inform. J.*, vol. 27, no. 1, 2021, Art. no. 1460458220982640.
- [19] O. Hoehne, "Rail systems viewed from a system-of-systems perspective," *INSIGHT*, vol. 19, no. 3, pp. 36–38, 2016.
- [20] C. Kotronis et al., "Employing SysML to model and explore levels-of-service: The case of passenger comfort in railway transportation systems," *Syst. Eng.*, vol. 23, no. 1, pp. 82–99, 2020.
- [21] P. M. Laso, D. Brosset, and J. Puentes, "Monitoring approach of cyber-physical systems by quality measures," in *Proc. Int. Conf. Sensor Syst. Softw.*, Springer, 2016, pp. 105–117.
- [22] D. Mourtzis et al., "Cloud-based cyber-physical systems and quality of services," *TQM J.*, vol. 28, no. 5, pp. 704–733, 2016.
- [23] M. Hobday, A. Davies, and A. Prencipe, "Systems integration: A core capability of the modern corporation," *Ind. Corporate Change*, vol. 14, no. 6, pp. 1109–1143, 2005.
- [24] J. M. Brooks et al., "Dueling stakeholders and dual-hatted systems engineers: Engineering challenges, capabilities, and skills in government infrastructure technology projects," *IEEE Trans. Eng. Manag.*, vol. 58, no. 3, pp. 589–601, Aug. 2011.
- [25] J. Whyte and A. Davies, "Reframing systems integration: A process perspective on projects," *Project Manage. J.*, vol. 52, no. 3, pp. 237–249, 2021.
- [26] C. Zheng et al., "Multidisciplinary integration during conceptual design process: A survey on design methods of cyber-physical systems," in *Proc. DS 84: Proc. DESIGN 2016 14th Int. Des. Conf.*, 2016, pp. 1625–1634.
- [27] R. Madachy and D. Jacques, "System cost modeling and SysML integration in model-based systems engineering," in *Proc. INCOSE San Diego Chapter Meeting*, 2017.
- [28] X. Pan et al., "Modeling and Simulation for Sos Based on the Dodaf Framework," in *Proc. IEEE 9th Int. Conf. Rel., Maintainability Safety*, 2011, pp. 1283–1287.
- [29] B. Boehm et al., *Cost Estimation With Cocomo II*. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.
- [30] B. Papke, S. Pavalkis, and G. Wang, "Enabling repeatable SE cost estimation with COSYSMO and MBSE," in *Proc. INCOSE Int. Symp.*, vol. 27, no. 1, 2017, pp. 1699–1713.
- [31] K. Post, G. Walley, and J. Che, "Integrating descriptive models with an analytical model culture—Lessons learned at Ford," in *Proc. INCOSE IW MBSE Workshop*, 2014.
- [32] O. Schönherr and O. Rose, "First steps towards a general SysML model for discrete processes in production systems," in *Proc. IEEE Winter Simul. Conf.*, 2009, pp. 1711–1718.
- [33] M. M. Hasan, P. Loucopoulos, and M. Nikolaidou, "Classification and qualitative analysis of non-functional requirements approaches," in *Enterprise, Business-Process and Information Systems Modeling*. Berlin, Heidelberg: Springer, 2014, pp. 348–362.
- [34] S. S. Alhir, *Guide to Applying the UML*. Berlin, Germany: Springer, 2006.
- [35] J. Holt and S. Perry, *SysML for Systems Engineering*, vol. 7, London, U.K.: IET, 2008.
- [36] C. A. Gonzalez Perez, M. Varmazyar, S. Nejati, L. Briand, and Y. Isasi, "A sysml-based methodology for model testing of cyber-physical systems," Univ. Luxembourg, Esch-sur-Alzette, Luxembourg, Tech. Rep. TR-SNT-2018-2, 2018.
- [37] D. Ameller et al., "Dealing with non-functional requirements in model-driven development: A survey," *IEEE Trans. Softw. Eng.*, vol. 47, no. 4, pp. 818–835, Apr. 2021.
- [38] E. Filippoulou et al., "Integrating cost analysis in the cloud: A sos approach," in *Proc. IEEE 11th Int. Conf. Innovations Inf. Technol.*, 2015, pp. 278–283.
- [39] M. Nikolaidou and C. Michalakelis, "Techno-economic analysis of SysML models," in *Proc. IEEE Int. Syst. Eng. Symp.*, 2017, pp. 1–6.

- [40] P. Leserf, P. de Saqui-Sannes, and J. Hugues, "Trade-off analysis for SysML models using decision points and CSPs," *Softw. Syst. Model.*, vol. 18, no. 6, pp. 3265–3281, 2019.
- [41] A. van Velzen et al., "Proposing a more comprehensive future total cost of ownership estimation framework for electric vehicles," *Energy Policy*, vol. 129, pp. 1034–1046, 2019.
- [42] R. Moyer, J. McGuigan, R. Rao, and W. Kretlow, *Contemporary Financial Management*. Scarborough, ON, Canada: Nelson Education, 2012.
- [43] *OMG SysML Specification Version 1.5*, OMG SysML, Milford, MA, USA, May 2017. [Online]. Available: <http://www.omg.org/spec/SysML/1.5/PDF>
- [44] B. A. Martens, "Costing of cloud computing services: A total cost of ownership approach," in *Proc. IEEE 45th Int. Conf. Syst. Sci.*, 2012, pp. 1563–1572.
- [45] V. Sower and C. Sower, *Better Business Decisions Using Cost Modeling*. New York, NY, USA: Business Expert Press, 2015.
- [46] K. Monisha and M. R. Babu, "A novel framework for healthcare monitoring system through cyber-physical system," in *Internet of Things and Personalized Healthcare Systems*. Berlin, Germany: Springer, 2019, pp. 21–36.
- [47] M. M. Baig and H. Gholamhosseini, "Smart health monitoring systems: An overview of design and modeling," *J. Med. Syst.*, vol. 37, no. 2, 2013, Art. no. 9898.
- [48] R. Baheti and H. Gill, "Cyber-physical systems," *Impact Control Technol.*, vol. 12, no. 1, pp. 161–166, 2011.
- [49] A. M. Madni, M. Sievers, and C. C. Madni, "Adaptive cyber-physical-human systems: Exploiting cognitive modeling and machine learning in the control loop," *INSIGHT*, vol. 21, no. 3, pp. 87–93, 2018.
- [50] C. Merlo, A. A. Akle, A. Llarra, G. Terrasson, E. Villeneuve, and V. Pilmieri, "Proposal of a user-centred approach for CPS design: Pillbox case study," *IFAC-PapersOnLine*, vol. 51, no. 34, pp. 196–201, 2019.
- [51] H. Baali, H. Djelouat, A. Amira, and F. Bensaali, "Empowering technology enabled care using IoT and smart devices: A review," *IEEE Sensors J.*, vol. 18, no. 5, pp. 1790–1809, Mar. 2018.
- [52] C. Kotronis et al., "Evaluating internet of medical things (IOMT)-based systems from a human-centric perspective," *Internet Things*, vol. 8, 2019, Art. no. 100125.
- [53] A. Burns et al., "SHIMMER—A wireless sensor platform for noninvasive biomedical research," *IEEE Sensors J.*, vol. 10, no. 9, pp. 1527–1534, Sep. 2010.
- [54] J. Ivković et al., "Odroid-xu4 as a desktop PC and microcontroller development boards alternative," in *Proc. 6th Int. Conf. Technics Inform. Educ.*, 2016, pp. 439–444.
- [55] C. Kotronis et al., "A model-based approach for managing criticality requirements in e-health IoT systems," in *Proc. IEEE 13th Annu. Conf.*, 2018, pp. 60–67.
- [56] NoMagic, Inc, Allen, TX, USA, "Cameo systems modeler," 2019. [Online]. Available: <https://www.nomagic.com/products/cameo-systems-modeler>
- [57] C. Kotronis et al., "Leveraging quality of service and cost in cyber-physical systems design," in *Proc. Int. Conf. Econ. Grids, Clouds, Syst., Serv.*, 2019, pp. 208–217.
- [58] "Openmodelica," 2019. [Online]. Available: <https://openmodelica.org/>
- [59] ATTIKO METRO, S.A., Athens, Greece, "Attiko metro," 2018. [Online]. Available: <http://www.ametro.gr/page/default.asp?id=4&la=2>
- [60] Urban Rail Transport, S.A., Athens, Greece, "Athens-piraeus electric railways," 2019. [Online]. Available: <https://bit.ly/3y9gg6A>
- [61] *CEN 13816:2002 Transportation—Logistics & Services—Public Passenger Transport—Service Quality Definition Targeting and Measurement*, EN Standard, 2002.



Christos Kotronis received the B.Sc. and the M.Sc. degrees in web engineering from the Department of Informatics and Telematics, Harokopio University of Athens, Greece, in 2014 and 2016, respectively.

He is currently a Postdoctorate Researcher with the Department of Informatics and Telematics, Harokopio University of Athens, Kallithea, Greece, under the supervision of Prof. Mara Nikolaidou.

From 2017 to 2019, Dr. Kotronis was the recipient of a three-year Engineering and Technology Sciences scholarship from the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI) for conducting his doctoral research.



Mara Nikolaidou (Member, IEEE) received the bachelor's and Ph.D. degrees from the Department of Informatics, University of Athens, Greece, in 1990 and 1995, respectively.

She is a Professor with the Department of Informatics and Telematics, Harokopio University of Athens, Kallithea, Greece. She has authored or coauthored more than 200 papers in international journals and conferences and actively participates in the organization of international conferences in the area of software and systems engineering. Her research interests include SoS engineering, system modeling and simulation, service-oriented architectures, and BPM. Over the last years she actively participated in numerous research projects funded by national, European and international agencies on system engineering, service-oriented architectures, digital libraries and e-government. She is currently involved in research projects focusing on SoS engineering, the Internet of Things, Cyber-physical Systems and Smart Cities.

Dr. Nikolaidou is a member of IEEE SMC society and Systems Council.



Anargyros Tsadimas received the B.Sc. degree in applied informatics from the University of Macedonia, Thessaloniki, Greece, in 2002, the M.Sc. degree in advanced information systems from the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Zografou, Greece, in 2005, and the Ph.D. degree in information systems with a focus on model-based design of enterprise information systems using SysML, from the School of Digital Technology, Harokopio University, Kallithea, Greece, in 2018.

He is currently a Teaching Laboratory Staff with the Department of Informatics and Telematics, Harokopio University. He has taught lessons like Operation Systems, Distributed Systems, System Analysis and DevOps at the Dept. of Informatics and Telematics of Harokopio University. He has 30 publications in international conference proceedings and Journals.



Christos Michalakelis received the Ph.D. degree in techno-economic analysis of telecommunications networks from the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece, in 2009.

He is an Associate Professor with the Department of Informatics and Telematics, Harokopio University of Athens, Kallithea, Greece. He has worked for many years with the Greek Ministry of Education, as an IT Manager. He has participated in a number of projects regarding the design and implementation of database systems, as well as in several technoeconomic and socioeconomic activities for telecommunications, networks and services. He is co-founder of "Study in Greece" (<http://www.studyingreece.edu.gr>), the official portal and initiative of Greece (Hellas), for the promotion and support of studies and educational activities in Greece for international students, acting as a cultural an educational bridge between Greece and other countries. He has authored or coauthored more than 100 papers to international journals and conferences. His research interests and field of expertise focus on technoeconomics engineering, costing, pricing and brokering services in the area of ICT, mainly cloud computing and the Internet of Things (IoT).



Dimosthenis Anagnostopoulos received the B.Sc. and Ph.D. degrees from the Department of Informatics, University of Athens, Greece, in 1991 and 1996, respectively.

He is a Professor with the Department of Informatics and Telematics in Information Systems and Simulation, Harokopio University of Athens, Kallithea, Greece. He currently serves as Secretary General for Information Systems of Public Administration in the Ministry of Digital Governance. He served as Rector of Harokopio University, from September 2011 to January 2016, and Dean of Faculty of Digital Technology, from 2016 to 2019. He is a visiting Professor with Sussex University, U.K. His research interests include Information Systems, eGovernment, Semantic Web and Web Services, Modeling and Simulation, Business Process Modeling.

Dr. Anagnostopoulos is an Associate Editor for *Requirements Engineering* (Springer) and served as the National Representative of Greece for ICT in Horizon 2020, from 2014 to 2015. He was elected Chair of the Greek Rectors' Conference (1/2014-6/2014).