Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 28 (2014) 481 – 488

Procedia Computer Science

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

# Reusable derivation of operational metrics for architectural optimization

Michael Masin[a]*, Henry Broodney[a], Candace Brown[b], Lior Limonad[a], Nir Mashkif[a], and Aviad Sela[a]

[a] *IBM Research, Haifa, Israel*
[b] *Lockheed Martin, West Palm Beach, U.S.A.*

**Abstract**

Maintaining coherence between system functional, performance, production and operational requirements is a key to the ability to optimize the design of large-scale systems. Different architectural configurations entail significant differences in functionality, performance, ease of manufacturing/assembly and operational behavior. While the first two are the usual concerns in architectural tradeoff analysis, the last two, reflected by manufacturability and operational metrics, such as manufacturability and affordability, are often neglected in architectural optimization. In this work, we propose a methodology to derive the formal specification of operational metrics applicable to design optimization based on life cycle processes, such as "manufacturing", "sunny day operation", and "unplanned maintenance". These operational metrics are presented in the context of an industrial case study for an Unmanned Underwater Vehicle (UUV) to provide context for the recommended approach. We suggest an approach to (1) define libraries of reusable operational metrics based on architectural properties, (2) build reusable data processing patterns to calculate these architectural properties, and (3) map calculated architectural parameters to a specific design model.



*Keywords*: Design Space Exploration; Operational Metrics; Operational Data; Black Box Integration, Reusable Data Patterns

* Corresponding author. Tel.: +972-4-8281369; fax: +972-4-8296115.
*E-mail address:* michaelm@il.ibm.com


doi:10.1016/j.procs.2014.03.059

## 1. Introduction

Maintaining coherence between functional, performance, production and operational requirements of a system is key to be able to optimize the design of large-scale systems. Different architectural configurations entail significant differences in functionality, performance, ease of manufacturing/assembly and operational behavior. While the first two are the usual concerns in architectural tradeoff analysis, the last two, reflected by manufacturability metrics, such as production cost and time-to-market, and operational metrics, such as life cycle cost and life cycle availability, are often neglected in architectural optimization. These usually drive total lifecycle cost and neglecting them in the early architectural design phase leads to cost overruns and subsequent project failures. To date, imposing operational constraints of this sort over Design Space Exploration (DSE) was often guided by simplistic expert estimations. Such manual estimates are not only hard to maintain, but, more crucially, may lead to erroneous design choices.

In this work, we propose a methodology for deriving the specification for reusable operational metrics, applicable to design optimization and based on operational life cycle processes, such as “manufacturing”, “sunny day operation”, and “unplanned maintenance”, and connecting them with historical data for metrics’ computation. However, using such historical data for DSE is still cumbersome since: (1) it corresponds to a specific design, (2) the data typically exist at the activity level, while it should be connected to metrics at system level, and (3) corresponding system parameters, such as resources’ costs or failure rates, should be aggregated in an appropriate level of abstraction according to the chosen optimization scope and architecture decision variables. For this purpose, we propose data processing patterns similar to the definition of the operational metrics.

Based on an industrial use case of an Unmanned Underwater Vehicle (UUV), we suggest an approach that (1) enables building of libraries of reusable operational metrics based on architectural parameters, (2) defines reusable data processing patterns to calculate these architectural parameters, and (3) maps calculated architectural parameters to a specific design model. The paper is organized in the following manner. In Section 2 we describe the logical architecture and life cycle processes of the UUV. Our methodology is based on Concise Modeling [1] and Pluggable Analysis Viewpoints [2] approaches to DSE and is briefly described in Section 3. In Section 4 the UUV use case is explored leading to the methodology discussion in Section 5 that concludes our findings. Section 6 briefly summarizes the work and outlines directions for the future research.

## 2. Unmanned Underwater Vehicle (UUV) use case description

UUV’s have been employed for decades to access the subsea environment as a safer and more economical alternative to manned vehicles and divers. The opportunities for application of UUV’s continue to expand as advances in autonomy and energy systems increase the intelligence and endurance of UUV’s. Standard missions include inspecting offshore installations such as oil rigs and pipelines, as well as conducting seafloor surveys to identify appropriate sites for planned subsea installations.

Lockheed Martin’s Marlin® Autonomous Underwater Vehicle, or AUV, is an example of an operational UUV that carries a 3D sonar payload to perform acoustic surveys of offshore platforms for the oil and gas industry. Marlin® provides the capability to autonomously perform post-construction and decommissioning surveys, as well as damage assessment after a hurricane. Fig 1(a) and 1(b) below show the Marlin® vehicle and an example of acoustic survey imagery that the onboard 3D sonar can provide. The processes and metrics described in this paper can be used to evaluate cost drivers for development and operation of a UUV like the Marlin®, to allow developers to make intelligent design decisions that result in an operationally affordable vehicle.

A common logical architecture is defined for UUV’s to include all of the fundamental subsystems required for the vehicle to operate. Fig. 2(a) below shows the architecture model which describes the logical structure for the key subsystems of a UUV. The payload is defined external to the vehicle because the payload is the reason for the vehicle’s existence- the vehicle acts as a “truck” to carry the payload and perform its mission. In addition to the 3D sonar payload that the Marlin® carries, other examples of payloads could be a sonar used to perform a seafloor survey, or an acoustic marker that a vehicle might deploy at a seafloor site to mark a location.

The vehicle logical subsystems are required to enable the vehicle to carry the payload and perform its mission. Sensors are required for the UUV to sense the characteristics of the environment in which it is operating, such as

water velocity or temperature. Flotation enables the UUV to be neutrally buoyant. A control system allows the UUV to maneuver and react to environmental disturbances such as waves or current. The energy system supplies power to the vehicle for the duration of its mission. The hull provides a hydrodynamic shape to the vehicle and protection for its internal components against debris. The propulsion system enables the vehicle to maintain forward motion. Structure provides a frame for all subsystems to be mounted and supports those subsystems under various structural loads caused by the vehicle's operations in the environment.

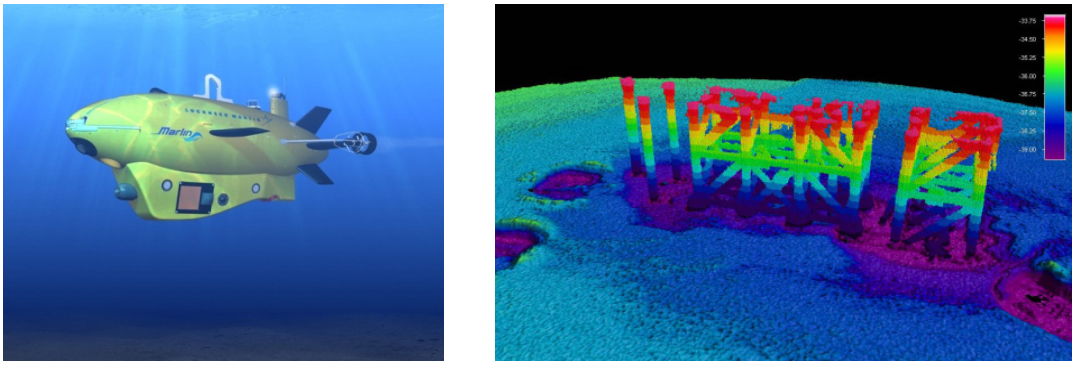

Figure 1 – (a) Marlin® AUV, (b) Acoustic Survey Imagery

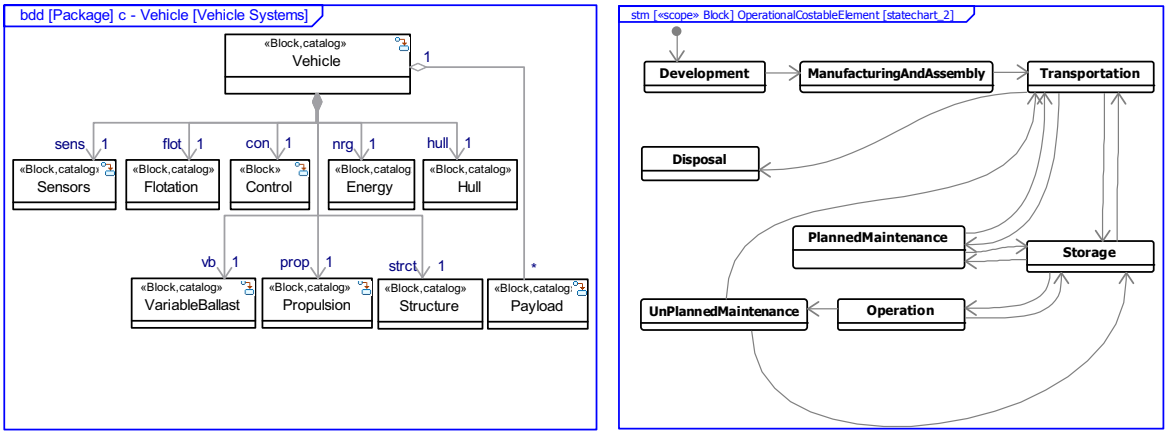


Figure 2 – (a) UUV Architecture, (b) UUV life cycle processes

Operational processes are usually defined hierarchically according to the required level of abstraction for current analysis. Fig. 2(b) provides a high-level definition of lifecycle operations for a UUV which could be further decomposed as necessary to account for operational analysis. According to this lifecycle the UUV starts in the Development state, which is the state in which the vehicle is conceptualized and designed. It then enters the Manufacturing and Assembly state in which it is fabricated, tested, and approved for use. At that point it begins

normal operational states in which it cycles between Transportation, Storage, Operation, and Planned and Unplanned Maintenance. Transportation defines the state the UUV is in while it is relocated between other states. Storage is the state in which the vehicle remains while it is not in use or being maintained. Operation describes the state in which the vehicle is performing its intended subsea mission. If a failure occurs during Operation then the vehicle may enter the Unplanned Maintenance state that it can be repaired. When the vehicle is no longer to be used it will enter the Disposal state and is removed from service.

## 3. Background: Concise Modeling and Pluggable Analysis Viewpoints

The proposed methodology is based on two preliminary works. The first is the *Concise Modeling* [1] extension of SysML [3] to specify architectural alternatives and system constraints. Concise modeling combines regular SysML diagrams defining architectural topology (e.g. components multiplicity and connections rules), with associated data tables called *catalogs* for blocks subclasses and *inventories* for parts multiplicities. DSE using concise modeling assumes that all parts are *a-priori* optional and divides all attributes into either *parameters*, determined by data tables or SysML model values, or *variables* optimized during DSE. SysML blocks like those illustrated in Fig 2(a) can be extended to have catalogs of alternative values. For example, when outlined by *catalog* stereotype, the energy block shown above has an associated catalog table listing several energy options and their relevant parameters (energy density, capital cost per kWh, fabrication time, etc.). These options could include batteries of various chemistries, fuel cells with various hydrogen and oxygen sources, or even a diesel engine with diesel fuel and an air source. All are possible selections for a UUV energy system.

The optimization process is responsible for finding concrete architecture alternatives which conform to the topology, driven also by a predefined set of system or subsystem attributes called *design objectives* and constraints grouped by different types of analysis, called *analysis viewpoints*. Analysis viewpoints help calculate the system's variables or define their feasible region for architectural optimization, considering concerns such as cost, weight, reliability, timing, resource allocation, and power and data distribution. For instance, in the case of the UUV energy system, the preferred energy option depends on the functional requirement to minimize the vehicle's weight while maximizing its endurance.

The second preliminary work [2] defines the concept of *Pluggable Analysis Viewpoints*, demonstrating its ability to specify design objectives as a library of reusable assets. The pluggable viewpoints are based on the concept of *Classification-by-Property* where the computational semantics for the sets appearing in the various analysis formulae is specified according to different properties of the domain elements. A designated API has been adopted from [4] to enable such property-based binding. Masin et al. [2] demonstrates that applying classification-by-property principles creates robust analysis libraries that remain resilient to system evolution during product design, and enables Lego™ type interoperability between different system analyses. In this approach, every domain defines a set of properties used for system analysis, such as component's *failure probability* for reliability analysis. One of the core API operators is **Scope**(*A*), which is interpreted as all domain elements that possess the set of attributes given by the argument *A*. For example, in order to calculate the *totalCost* of the UUV described in Fig 2(a), one may sum the cost of all UUV elements that feature a 'cost' attribute and are contained within the "vehicle" block.

In this work we show how the same modeling approach may be enhanced to particularly take into account the *operational objectives* depicted from operational models such as Fig. 2(b), evaluate architectural parameters using operational or simulated data and combine all in one integrated SysML model.

## 4. UUV Operational Metrics

### *4.1. Process based metrics*

In our example, the Systems Engineer is interested in the affordability of an unmanned underwater vehicle, which can be characterized by two objectives: minimizing total lifecycle-cost and maximizing availability. The following equation demonstrates how total Life Cycle Cost (LCC) can be computed as a simple sum over the Net Present Value of costs of all lifecycle processes:

(1) $LifeCycleCost[s] = DevelopmentCost[s] + ManufacturingAndAssemblyCost[s] + TransportationCost[s] + StorageCost[s] + OperationCost[s] + PlannedMaintenanceCost[s] + UnPlannedMaintenanceCost[s] + DisposalCost[s]$
$\forall\, s \in \boldsymbol{Scope}(LCC, DevelopmentCost, ManufacturingAndAssemblyCost, TransportationCost, StorageCost, OperationCost, PlannedMaintenanceCost, UnPlannedMaintenanceCost, DisposalCost)$

Fig. 3 (a) suggests a Parametric Diagram for calculating *LifeCycleCost*. Similarly, availability can be computed as the percentage of operational time that the system is in a phase to be capable of executing a mission:

(2) $TotalAvailability[s] = \frac{OperationTime[s]+StorageTime[s]}{OperationTime[s]+StorageTime[s]+TransportationTime[s]+PlannedMaintenanceTime[s]+UnPlannedMaintenanceTime[s]}$
$\forall\, s \in Scope(TotalAvailability, OperationTime, StorageTime, TransportationTime, PlannedMaintenanceTime, UnPlannedMaintenanceTime)$

This enables maintaining the specification of operational objectives independent of the operational models specifications. In this discussion we leave out of scope detailed description of these semantic mappings, the interested reader can refer to article [2]. The next step is to assign concrete metrics to each of the operational measures. These metrics are coming from architectural model attributes.

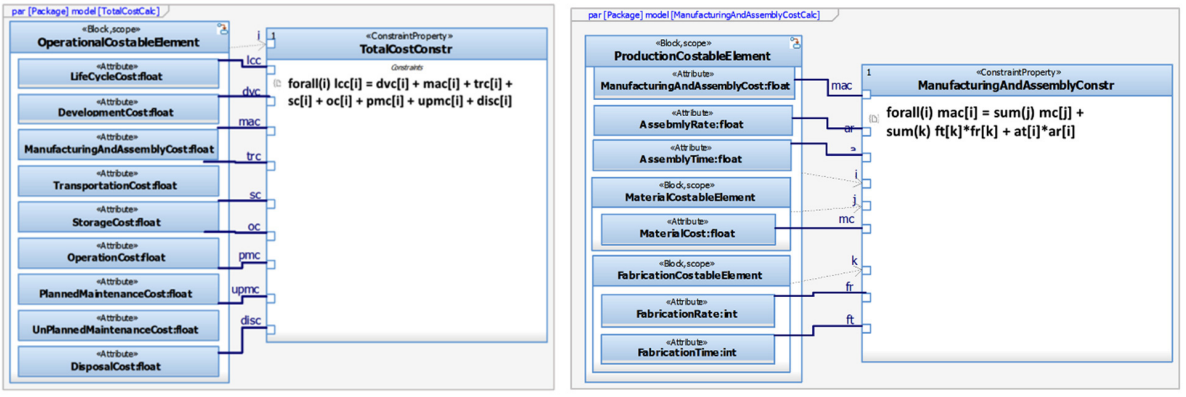


Figure 3 – (a) Process based LCC, (b) Architecture based manufacturing and assembly cost

### *4.2. Architecture based metrics*

The next methodological step is specifying a metric for each of the measures associated with some activity in the operational model. Using [2] methodology, each metric is defined by its respective ontological concepts – attributes. For example, the *ManufacturingAndAssemblyCost* may be specified as follows:

(3) $ManufacturingAndAssemblyCost[self] =$
$\sum_{s \in \boldsymbol{Scope}(MaterialCost, isContained[self])} MaterialCost[s] +$
$\sum_{s \in \boldsymbol{Scope}(FabricationRate, FabricationTime, isContained[self])} FabricationRate[s] \cdot FabricationTime[s] +$
$AssemblyRate[self] \cdot AssemblyTime[self]$
$\forall\, self \in Scope(ManufacturingAndAssemblyCost, AssemblyRate, AssemblyTime)$

As shown in Fig 3(b) using Parametric Diagram notation, this metric depends on five attribute types. Three are attributes of each of the contained subsystems: material cost, manufacturing rate, and manufacturing time. The remaining two are attributes of the system itself: assembly rate, and assembly time. The block elements are classification-by-property blocks (expressed by the *scope* stereotype) that define sets of all elements that have the specified attributes. The metrics are completely detached from the specific design model and can be freely reused. On the other hand, when these parametric diagrams are added to a specific model, the blocks could be used to add properties to the underlying SysML model. For example, if the Vehicle block from Fig 2(b) is generalized from *ProductionCostableElement*, then the Vehicle block inherits *ManufacturingAndAssemblyCost*, *AssemblyTime* and *AssemblyRate* attributes.

### *4.3. Evaluation of architectural parameters and connection to the UUV model*

Metric's ontology defines the system attributes used for its calculation. When possible, these attributes should be evaluated based on historical data or simulations. In this section we show how using an approach similar to pluggable analysis viewpoints we can define reusable data processing patterns. Let us consider evaluation of the *FabricationTime* required for calculation of *ManufacturingAndAssemblyCost*. Fig. 4 (a) shows how an intermediate value of *PartFabricationTime* is calculated based on operational data and then aggregated for each subsystem. The *new* stereotype is used when new queries or attributes are defined that should *not* be part of Scope in the definition of the set of elements. *aggregateByValue* stereotype is used for aggregating transactions according to the attribute value. In this example it is used to summarize the *FabricationTime* per part and then separate it per subsystem.

The generic metric and evaluated parameters should be connected to the specific architectural model, UUV in our case. In Fig. 4 (b) we show steps, denoted by numbers on arrows, how Concise Modeling catalog tables of UUV subsystems could be filled by the external table of UUV production data. They should be synchronized using system and subsystem ID's or mapped between the metric's and model's ontologies. Adding relevant attributes to UUV model blocks is enabled by adding aggregation relations between each of the UUV blocks and the metrics blocks. In Fig 4 (b) we illustrate the aggregation relations between the UUV blocks and the *ProductionCostableElement* block, enabling the UUV blocks to inherit the *FabricationTime* attribute.

### *4.4. Additional considerations*

When operational data does not exist for all potential system architectures or it is impossible to run simulations for all potential architectures, a statistical model should be built to evaluate system metrics. For example, a response frontier could be built for the legacy system simulation to estimate operational cost for all potential architectures. Statistical models become especially critical for exploring large design spaces. When a statistical model is used instead of regular metric defined in Section 4.2, it should be brought to the system model including specification of its parameters. The parameters could be imported to the SysML model using the external data approach of Section 4.3. Additional constraints should be added for meaningful design solutions, e.g., to account for incompatible combination of decision variables of the statistical model. Statistical models should usually be used for parameters evaluation instead of the direct procedure shown is Section 4.3; in this case, data processing patterns bring data to the statistical model, using the same approach, instead of bringing the data directly to the SysML model.

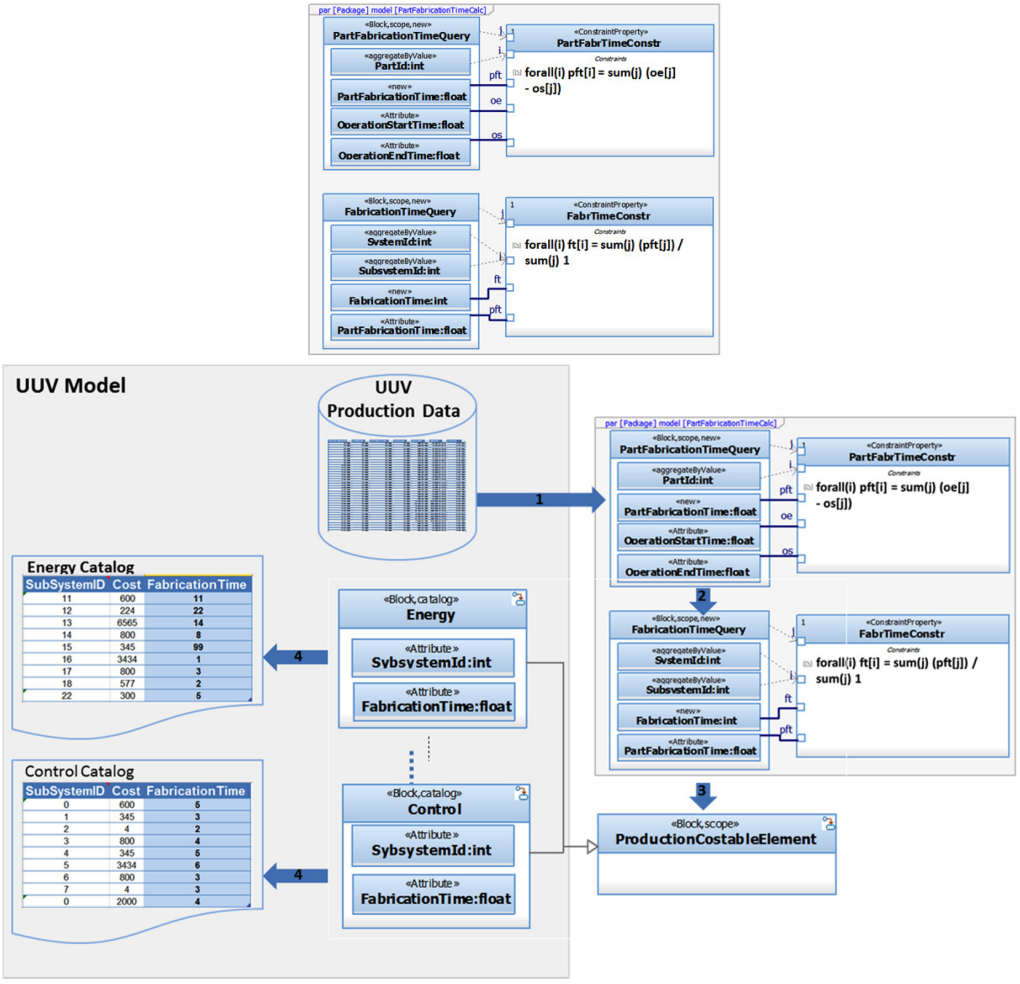


Figure 4 – (a) Evaluation of *FabricationTime*, (b) Data integration flow

Metric definition of Section 4.1 requires all cost and availability formula components to be defined, which could be a corporate policy for calculation of affordability. If all cost components are optional then the equations should be changed accordingly using sum over Scope of each optional attribute. When existing methodologies and software for Life Cycle Cost Accounting (LCCA) [5], e.g., required by DoD and DoE, as well as CoCoMo / CoSysMo / CoQualMo [6,7,8,9,10] cost models are available, they could be directly incorporated for definition of metrics and evaluation of relevant parameters.

## 5. Concluded Methodology

Following the methodological lesson learned by the authors in the course of modeling and design optimization of the illustrated UUV system, we generalize it for the common case of a complex system design. The following steps have been identified:

0. Prepare libraries of operational metrics and life cycle process models.
1. Apply all appropriate operational metrics from libraries. Complete additional metrics using libraries of life cycle processes (consider adding them to the library), using statistical models for large trade spaces.
2. Build data processing patterns for evaluation of all system parameters used in the previous step. Use statistical models when needed.
3. Build a system model for the architectural design, bringing relevant operational metrics and mapping external data sources to the potential architectural elements catalogs.
4. Combine operational metrics with other system metrics and requirements to find the optimal solution(s).
5. Check the solution(s).Stop if satisfied. Otherwise update the statistical model and other system metrics and requirements and return to Step 4.

## 6. Summary

In this paper we extended our previous work on architectural DSE using concise modeling and pluggable analysis viewpoints to incorporate operational metrics. The work has been done in the context of an Unmanned Underwater Vehicle (UUV) use case. We leveraged our experience to outline a generic methodology for architectural design where operational metrics must be considered. This methodology can be used for building classification-by-property reusable operational metrics libraries, detached of a specific design problem, and reusable data processing patterns for evaluation of metrics, including Parametric Diagrams visualization. We also show how to integrate the external data sources in concise modeling. In ongoing research we will complete the formal definition of the metrics definition language, and complete the development of efficient statistical models for DSE and data processing.

## References

[1] H. Broodney, D. Dotan, L. Greenberg, and M. Masin, “Generic Approach for Systems Design Optimization in MBSE,” in *The 22nd Annual INCOSE International Symposium*, 2012.
[2] M. Masin, L. Limonad, A. Sela, D. Boaz, L. Greenberg, N. Mashkif, and R. Rinat, “Pluggable Analysis Viewpoints for Design Space Exploration,” *2013 Conference on Systems Engineering Research, Procedia Computer Science*, vol. 16, no. 0, pp. 226–235, 2013.
[3] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language*. 2008, p. 576.
[4] J. Parsons and Y. Wand, “Emancipating instances from the tyranny of classes in information modeling,” *ACM Transactions on Database Systems*, vol. 25, no. 2, pp. 228–268, Jun. 2000.
[5] G. A. Norris, “Integrating life cycle cost analysis and LCA,” *International Journal of Life Cycle Assessment*, vol. 6, pp. 118–120, 2001.
[6] “Cost Structure and Life Cycle Cost (LCC) for Military Systems,” in *Papers presented at the RTO Studies, Analysis and Simulation Panel (SAS) Symposium*, 2001.
[7] C. Kemerer, “An empirical validation of software cost estimation models,” *Communications of the ACM*, vol. 30, 1987.
[8] R. Valerdi, B. Boehm, and D. Reifer, “COSYSMO: A constructive systems engineering cost model coming of age,” in *Proceedings of the 13th INCOSE Symposium*, 2003.
[9] R. Madachy and B. Boehm, *Assessing quality processes with ODC COQUALMO*. Lecture Notes in Computer Science, Making Globally Distributed Software Development a Success Story, 2008.
[10] J. Powell, “An early look at COQUALMO in the JPL environment,” *Methodology*, 2002.