Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/110106

Category Theory as a Formal Mathematical Foundation for Model-Based Systems Engineering

Mohamed A. Mabrok[1,2,*] *and Michael J. Ryan*[1]

[1] School of Engineering and Information Technology, University of New South Wales, Australian Defence Force Academy, Canberra ACT 2600, Australia.
[2] Robotics, Intelligent Systems and Control Lab, Computer, Electrical and Mathematical Science and Engineering, King Abdullah University of Science and Technology (KAUST), Saudi Arabia.

Received: 2 Sep. 2016, Revised: 15 Nov. 2016, Accepted: 22 Nov. 2016
Published online: 1 Jan. 2017

Abstract: In this paper, we introduce Category Theory as a formal foundation for model-based systems engineering. A generalised view of the system based on category theory is presented, where any system can be considered as a category. The objects of the category represent all the elements and components of the system and the arrows represent the relations between these components (objects). The relationship between these objects are the arrows or the morphisms in the category. The *Olog* is introduced as a formal language to describe a given real-world situation description and requirement writing. A simple example is provided.

Keywords: Category Theory, Model-based systems engineering

1 Introduction

Category Theory is a branch of mathematics, born from algebraic topology, and designed to describe various structural concepts from different mathematical fields. Category Theory provides a set of concepts and theorems that form an abstraction of a wide range of concrete concepts in many branches of mathematics, including computing science [1]. It is considered as the most general and abstract branch of pure mathematics [2].

Unlike set theory, Category Theory has fewer mathematical structures that govern the transformation between objects and focuses on the relationship between the elements not on the elements themselves, offering a pure theory of functions in themselves, not a theory of functions derived from sets [3]. Category Theory can be used used to formalize mathematics and its concepts as a collection of *objects* and *arrows*, sometimes called morphemics.

Category Theory has been used since the nineties in computer science, data science and software engineering; see for instance, [4,5,6,7,1,8,9,10,11,12]. However the use of Category Theory in engineering modelling is considered to be new. In a comprehensive unpublished report [13], the author describes how engineering models, as they are constructed for manufactured products and biomedicine, can be embedded as axiom sets using the general concepts of Category Theory. Also, in a recent study that discusses a formal foundation of systems engineering based on Category Theory presented in [14] shows that Category Theory can be adapted to current modelling tools and languages (e.g. SysML).

System engineering (SE), as a field of research and study, is leading the effort to create strategies to cope with changes in technology and systems complexity in engineering and other fields. SE is an interdisciplinary effort that focuses on how to design and manage complex engineering systems over their life cycles. Issues such as requirements engineering, reliability, logistics, coordination of different teams, testing and evaluation, maintainability and the coordination of many other disciplines necessary for successful system development, design, implementation, and ultimate decommission become more difficult when dealing with large or complex projects. Systems engineering deals with work processes, optimization methods, and risk management tools in such projects.

A *system* in this perspective can be defined as a combination of interacting elements organized to achieve one or more stated objectives [15]. Bunge, in his

* Corresponding author e-mail: m.a.mabrok@gmail.com

development of general system theory (GST) [16], views the system as an object composed of several parts which have relationships between each other. In both views, the system is an object which allows different elements to be taken as a whole [17]. A system can therefore readily be considered as a *category*, in which the elements and components of the system are the objects (things) of the category and the relationships between the elements and the components are represented by the arrows.

In this paper, we present a generalised view of the system based on Category Theory. The paper introduces a view of any system as a category; where the objects of the category represent all the elements and components of the system and the arrows represent the relations between these components (objects). For instance, stakeholders, hardware components, software components and the environment systems that interact with the system of interest are all objects of the category that represent that system. The relationship between these objects are the arrows or the morphisms in the category. The use of Category Theory to describe a system is a form of Model-based system engineering (MBSE).

2 Model-based system engineering

The concept systems theory or general system theory (GST) goes back as early as the 1920s and 1930s to Bertalanffy, however, it was not widely known in the scientific community until 1950s [18]. Bertalanffy's goal was to put one heading to the organismic science that he had observed in his work as a biologist. He intends to use the word system for those principles that are common to systems in general. GST has developed in many fields such as systems biology, cybernetics, adaptive systems and systems engineering. Systems engineering in general can be considered as a particular view of the general systems theory that deals with engineering systems and techniques.

Model-based system engineering (MBSE) is focused on developing a set of architectures, tools and methodologies which significantly contributes in closing the gap between the customer needs and the subsequent development of systems, products, and services [19,20, 21,22,23]. The term MBSE was initially introduced by Wymore in 1993 [24], where he provided a rigid mathematical framework for MBSE for large-scale and complex systems.

2.1 Wymore's view of a MBSE

Wymore defines a discrete system as follows [24],;

Definition 1.*A discrete system is a quintuple* $Z = (SZ, IZ, OZ, NZ, RZ)$ *where,*

–*Z is the system under consideration*
–*SZ is the set of states of the discrete system Z,*
–*IZ is the set of inputs of the discrete system Z,*
–*OZ is the set of outputs of the discrete system Z,*
–*NZ is the set of next state function of the discrete system Z,*
–*RZ is the set of readout function of the discrete system Z.*

Wymore described the system model as comprising two separate parts; the interior part of the system, which is described as *states* and the exterior part which is described as *inputs/outputs*. From the exterior part, the system receives inputs to process through the states in the interior part of the system to deliver systems outputs. According to Wymore, a system design problem needs six categories of system requirements given as follows;

–Input / Output Requirements which specify what does the system do and what doesn't it do
–Performance Requirements which specify how well the input/output requirements shall be met.
–Technology Requirements
–Cost Requirements which specify what the system will cost to build and operate to satisfy the input/output.
–Tradeoff requirements which specify how well the performance requirements is to be traded off against the cost requirements.
–System Test Requirements which specify how can it be proven that the system meets input/output expectations?

Despite the effort that Wymore provided in his work, it is difficult to translate his mathematical framework to a real modelling tool. Perhaps, the main reason behind that is his focus on the high level behaviour of the system without giving attention to the physical aspect of the system and detailed design. In this paper, we use the a similar construct of inputs/outputs concept but in a form that can be used in the detailed design.

Wymore made extensive use of mathematical set theory as a basis for formalizing MBSE. However, many people have found that Wymore's book is hard to follow such that only mathematicians are able to read it. Some of these reviews can be found in Amazon on-line book store. Most of there reviews concluded that Wymore provides a rigorous mathematical foundation for systems engineering, however his work has been sadly neglected for most of that time because of mathematical difficulty of his book.

2.2 Model-based system engineering definition

The International Council on Systems Engineering (INCOSE) [15] defines MBSE as *the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases*. In

[25] MBSE is defined as the notion that we can construct a model of a system that we can transform into the real thing.

Based on INCOSE's definition, MBSE concerns the set of tools and methodologies that transform the system between several domains throughout the system life cycle. These domains start with customer needs and problem formulations. Then, customer needs are transformed into a formal set of system requirements, where a formal description of the problem is given. Next, design, analysis, verification and validation are considered as part of a transformation into a solution domain, where an optimal solution is chosen.

Mathematical based modelling is in the heart of technical engineering domains. Engineers have always built models for their systems of interest in order to investigate, improve, validate, and test their designs. However, this is not the case in the higher level engineering practices, for instance, in systems engineering. Supporting engineering modelling in a higher level with a mathematical formulation will allow us to use the computational power of the computers to optimize over the entire system. Also, having such model will allow a dynamical relationship between the design parameters (control parameters) and the system's requirements and constraints. Of course, the level of abstraction in the higher level modelling must be considered.

In general, a system model is defined as the conceptual description of a system that comprises multiple views such as planning, requirement (analysis), design, implementation, deployment, structure, behaviour, input data, and output data views [26]. Therefore, the model should be able to express all the relations between the system elements. Mathematically, this model description looks attractive, however, structural constraints that govern most of the mathematical theories limit the ability of these theories to provide a real word system model. For instance, set theory has a mathematically interesting structure, but that structure is one of the reasons that Wymore's theory is restricted. This implies that a logical, more abstract mathematical theory is needed in order to provide a foundation for system modelling based on mathematical logic and also applicable to the real world situations.

There are many reasons for the need of mathematically based foundation for MBSE. For instance, writing requirements, there is no agreed formula for writing a "good" requirement, which can be transformed or decomposed to a lower level in the design process. Traceability is another important reason for the need of a logical foundation for MBSE. A mathematically based foundation for MBSE will facilitate the verification and validation activities for the system to be modelled. In the next section, a brief description for Category Theory is provided.

3 Introduction to Category theory

Category Theory is a relatively recent branch of mathematics, stemming from algebraic topology, and designed to describe various structural concepts from different mathematical fields in a uniform way [1]. It is a general mathematical theory of structures and of systems of structures. It is considered to be in a central position in contemporary mathematics and theoretical computer science, and is also applied to mathematical physics. It is a conceptual framework, which allows visualizing the universal components of a family of structures of a given kind, and how structures of different kinds are interrelated.

Category Theory was invented by Samuel Eilenberg and Saunders Mac Lane in the early 1940s. It was specifically designed to bridge what may appear to be two quite different fields: topology and algebra [27]. However, many mathematicians saw Category Theory as a new foundation for all mathematical thought, and the theory has branched out into certain areas of sciences such as quantum physics and computer science. In addition, it is symbolically attractive as it has symbolism that allows the visualization of quite complicated facts by means of diagrams [28].

The study of categories is an attempt to capture axiomatically what is commonly found in various classes of related mathematical structures by relating them to the structure-preserving functions between them. A systematic study of category theory then allows the proof of general results about any of these types of mathematical structures from the axioms of a category. Category Theory has the ability to formalize concepts of other high-level abstractions theories such as set theory, ring theory, and group theory. Consequently, Category Theory can also be considered as a vehicle that allows problems from one area of mathematics to be transformed to another area, where solutions are sometimes easier [28].

From a mathematical perspective, a category is defined as an algebraic structure that contains *objects* (morphisms), which are connected by *arrows*. There are two main basic properties for a category; associativity and the existence of an identity arrow for each object. An example of a category, is the category of sets, where objects are sets and arrows are functions.

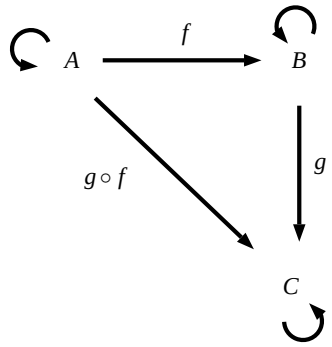


Fig. 1: This represent a category with three objects A,B,C, three arrows (morphisms) denoted $f,g,g\circ f$ and three loops are the identity arrows. The two main basic properties for a category are associativity and the existence of an identity arrow for each object.

In short, a category C is defined by a collection of:

–Objects: $ob(C)$ represents any thing.
–Arrows: represents the relationship between the objects, where, each arrow f has a unique source object A and target object B and A and B are in $ob(C)$.

The two basic axioms properties for a category are defined as follows;

–Identity: for any object $A\in ob(C)$; there exist an arrow $I_A : A\longrightarrow A$ normally given as ⟲
–Associativity: if $f : A\longrightarrow B$, $g : B\longrightarrow C$ and $h : C\longrightarrow D$; then $h\circ(g\circ f)=(h\circ g)\circ f$.

A formal definition for a category is given as follows [28];

Definition 2.*[28] A category is a quadruple $C=(ob(C),hom,id,\circ)$ consisting of:*

1. *a collection $ob(C)$, whose members are called C-objects,*
2. *for each pair (A,B) of C-objects, a set $hom(A,B)$, whose members are called C-morphisms from A to B [the statement $f\in hom(A,B)$ is expressed more graphically by using arrows; e.g., by statements such as $f : A\longrightarrow B$ is a morphism,*
3. *for each C-object A, a morphism A $I_A : A\longrightarrow A$, called the C-identity on A,*
4. *a composition law associating with each C-morphism $f : A\longrightarrow B$ and each C-morphism $g : B\longrightarrow C$ an C-morphism $g\circ f : A\longrightarrow C$ called the composite of f and g, subject to the following conditions:*
   - *(a)composition is associative; i.e., for morphisms $f : A\longrightarrow B$, $g : B\longrightarrow C$ and $h : C\longrightarrow D$; then $h\circ(g\circ f)=(h\circ g)\circ f$ holds,*
   - *(b)C-identities act as identities with respect to composition,*

By definition, an object in a category can be any thing, therefore, it is possible to have a category of categories. However, the arrows or the morphisms between categories in a category of categories are called *functors.*

Definition 3.*[28] Assume that C and $\overline{C}$ are categories and A and B are objects in the category C and there is a morphisms $f : A\longrightarrow B$. Then, a functor F from C to $\overline{C}$, $F : C\longrightarrow\overline{C}$, is a function that assigns to each C-object A a $\overline{C}$-object $F(A)$, and to each C-morphism $f : A\longrightarrow B$ a $\overline{C}$-morphism $F(f) : F(A)\longrightarrow F(B)$, in such a way that:*

1. *F preserves composition, and*
2. *F preserves identity morphisms.*

Note that, in the above definition, the functor $F : C\longrightarrow\overline{C}$ is technically a family of functions and transformations; see Fig 2.

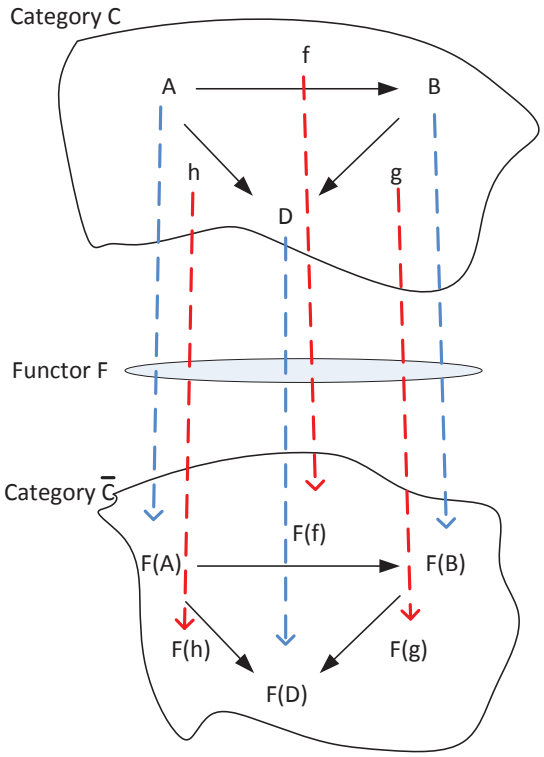


Fig. 2: Transformation or mapping in categorical language (functor) between two categories. The functor $F : C\longrightarrow\overline{C}$ is a family of functions, where the set of red dashed lines represent the transformation of the morphisms and the set of blue dashed lines represent the transformation of the objects.

One of the main advantages in Category Theory is the visualization of its objects and transformations between them [29].

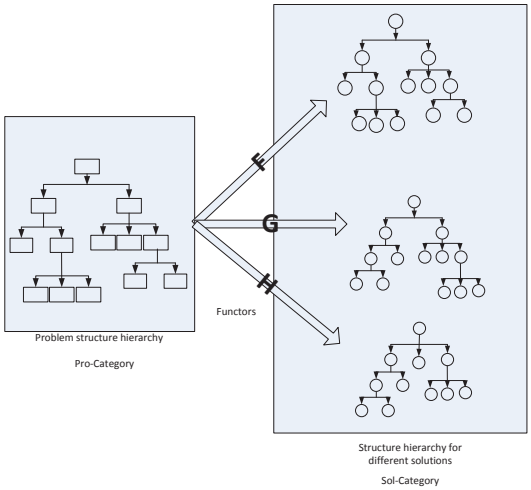


Fig. 3: Transformation or mapping in categorical language as a morphisms (functor) between two categories; the problem-category(*Pro*) and the solution-category (*Sol*).

4 Introduction to ologs

Olog is a term that was introduced by David Spivak in [30] which refers to ontology log, where ontology is the study of what something is, i.e the nature of a given subject, and ologs are designed to record the results of such a study [30]. The olog is a category that can be used model or formalize a given real-world situation. Ologs are similar slightly more sophisticated, mathematical form of semantic networks. According to [30], the main advantages of using an olog rather than writing a prose description of a subject are the following:

–an olog gives a precise formulation of a conceptual world-view,
–an olog can be formulaically converted into a database schema,
–an olog can be extended as new information is obtained,
–an olog written by one author can be easily and precisely referenced by others,
–an olog can be input into a computer and meaningfully stored
–different ologs can be compared by functors, which in turn generate automatic terminology translation systems.

An olog is a category where the objects are boxes with English-language phrases and the arrows represent relationship between these boxes. These relationship can be aspects, attributes or observables. In ologs, commutative diagrams represent facts.

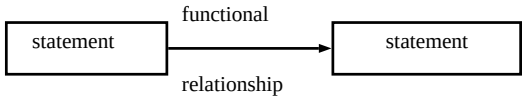


Fig. 4: The olog is a category where the objects are boxes with English statement and the arrows are functional relationships between these statements.

A ***type*** in olog is an abstract concept which represents a singular indefinite noun phrase. For example, in Fig. 5, every box represent a type a particular thing. In other words, a type represents a class of objects or individuals which share common characteristics.

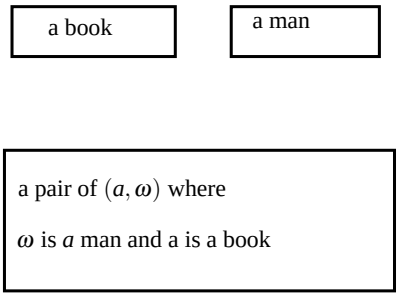
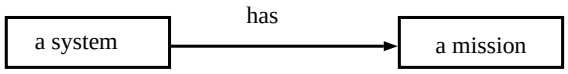


Fig. 5: Each of these boxes represent a type of a particular thing, i.e, a man represents a class not a particular man. Similarly, a book represents a class that contain all books not a particular book.

Aspects is another basic ologs concept, which represents a particular way of viewing a thing. For example, ”a book” has ”a title”, this implies that ”a title” is an aspect of the book. Also, ”a man” is ” a person” ; hence ”being a person” is an aspect of a man.

Suppose that we have an object called ”system”, one of its aspects is having a mission.



Importantly, aspects are functional relationships. Therefore, we must be able to draw an arrow from each dot in the ”system” set to a unique dot in the ”mission” set.

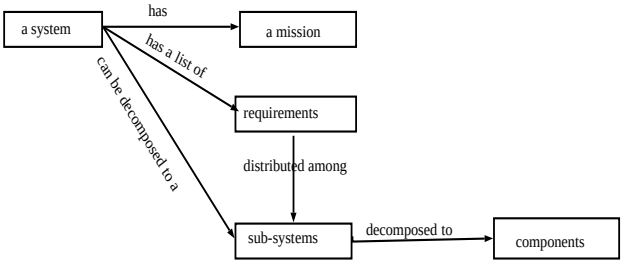
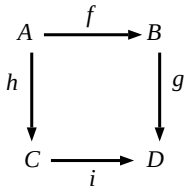


Fig. 6: An object can have many aspects, as for example, a system has a mission, a set of requirements and can be decomposed to sub-systems.

Facts is another concept in ologs, which represents the path equivalences. For instance, the bath ”a system has a list of requirements distributed among sub-systems” is equivalent to the bath ”a system can be decomposed to sub-systems”. This concept in category theory is called *commutative diagram*.



Instance in ologs represents a particular element from a type, i.e, each type can be assigned a set of instances. For example, a radar system is an instance of the type system.

5 System as a category

As noted earlier, if a system is defined as a set of interacting elements, then a system is readily able to be considered as a category; where the objects of the category represent all the elements of the system. The interactions or the relations between these objects can be represented by the arrows or morphisms. In addition, since an object can be a category itself, a system of systems can be also seen as a category of categories.

From a system design perspective, a requirement or a set of requirements can be considered as an object. Further, a physical element that satisfies a requirement is also an object. The mapping or the transformation between the requirements and the corresponding physical components can be viewed as arrows or morphisms in a categoral sense. This view of a system as a category will facilitate the traceability of the system requirements. Also, the path equivalence concept in category theory will facilitate the consideration of several design alternatives.

System attributes or quality attributes of a system which are used to evaluate the performance of a system, sometimes called system ”ilities” such as accessibility, affordability, availability, reconfigurability, ect..., can be also modelled as objects in the category. The relationship between these attributes and the system’s components and elements and also the degree of satisfaction for each attribute can be described using the arrows or the morphisms.

To be more precise, let us define a new category **Sys,** where the objects are the elements of the system of interest. An element or object can be a descriptive sentence (such as requirements, or system specification), or, it could be an element such as hardware component, physical part, or an individual person (an employee for example).

Model-based system design is a major part of MBSE. Generally, the system design process is a mapping process from the problem domain to solution domain. Using systems engineering language, the design process can be viewed as transforming or mapping the requirements from the problem domain into design parameters and components in a solution domain. This transformation or mapping can be expressed in categorical language as a morphisms (functor) between two categories; the problem-category(*Pro*) and the solution-category (*Sol*). Definition 3 defines the mapping between two categories and the conditions that are needed to be preserved under this mapping. The *Sol*-category can contain different configurations or solutions, where every solution is a result of different functor as illustrated in Fig. 3.

An olog provides a rigorous mathematical framework for knowledge representation and construction of engineering models. It can be of a great help in formalizing MBSE and model-based design. For instance, in requirements engineering, an olog can be used as a formal semantic to write a requirement statement or statement in general. For example, an olog for system functional requirement can be modelled as in Fig.7.

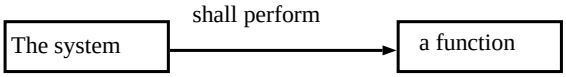


Fig. 7: Using an olog to represent simple requirement statement.

In model-based design, an olog can be used to generate new ideas and design support by integrating new and old knowledge. Also, the path equivalence property can be used to compare different design solutions for a particular requirement. Furthermore, the visualization of Category Theory in general and ologs in particular also helps in building the system architecture using the boxes as objects and arrows as relations.

Since the problem description in *Pro*-category is almost descriptive terms; i.e., statements such as requirements and constraints, it is convenient to use the ologs introduced in Section 4. In fact, ologs provide a formalized way of writing an English statement. Then, this object (olog) can be parametrized and mapped from *Pro*-category to *Sol*-category where solution elements are generated.

One useful property of the category theory is the duality principle for categories, which states: *Whenever a property P holds for all categories, then the reverse of that property holds for all categories.* This duality principle can be used to conduct the validation and verification of the designed system.

Consider a design problem of a rainwater supply system to a residential town. In this example, the problem definition can be stated as follows.

The average rainfall per year in the town is approximately 633*mm*. The population of the town is 4000 residents in 1000 houses. A a robust rainwater supply system is required to maintain the rainwater as the primary source of the water for the town and make it accessible and available all year.

The main three requirements of this problem can be stated as follows;

- R_1 : The system shall collect the rainwater that falls in the area.
- R_2: The system shall store the rainwater in a safe and accessible place.
- R_3: The system shall redistribute the rainwater to the houses.

With the two main constraints (aspects):

- Cs_1: The system must sustain the water supply to each house over all the year at the rate of y litres/day.
- CF_1: The total system cost must be minimized.

These requirements and aspects can be modelled as in Fig. 8 using an ologs.

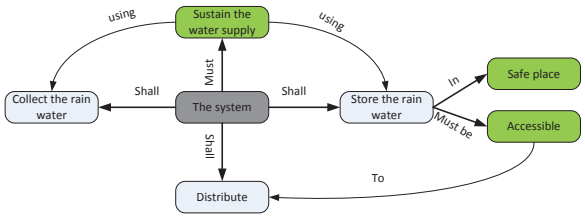


Fig. 8: Modelling the requirements and attributes of water system design using olog.

The olog in Fig. 8 represent the main functions that the system should perform (collect, store and distribute). Also, we can add *aspects* or attributes to the system such as (sustain the water supply) or to each functional requirements such as (stored in safe accessible place). Any suggested design must perform the given functions and address the system's aspects.

The different conceptual designs for the water system can be viewed as an *instances* from the olog in Fig. 8. Where every instance represent a particular design. For example, suppose that in the conceptual design phase, two basic classes of solutions (*instances*) are proposed as following;

1. Solution 1: Central dam which collects the rainwater from a rainfall catchment area, then filters it and redistributes the filtered water directly to the houses; see Fig 9.

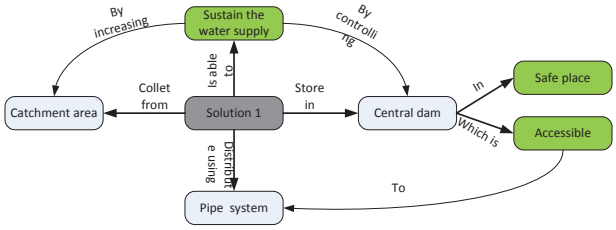


Fig. 9: Conceptual design for solution 1 for the water system, which can be viewed as an *instance* from the olog in Fig. 8.

2. Solution 2: Distributed storage system, where each house has its own storage unit (water tank for instance) to store rainwater collected from its own roof; see Fig 10.

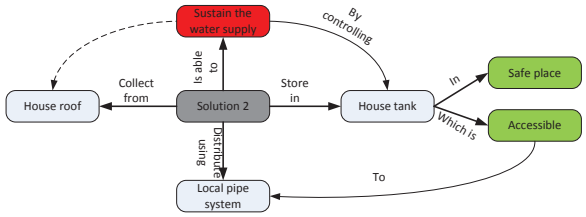


Fig. 10: Conceptual design for solution 2 for the water system, which can be viewed as an *instance* from the olog in Fig. 8.

Another conceptual design can be generated from the previous designs, which can be given as follows;

3. Solution 3: Area storage system, where every suburb in the town has one central storage system.

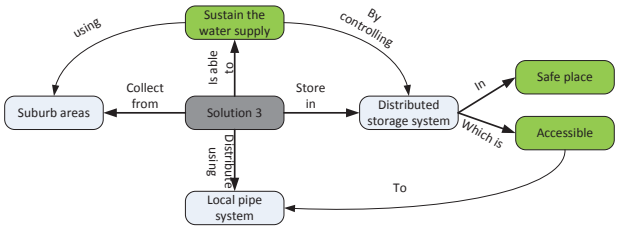


Fig. 11: Conceptual design for solution 3 for the water system, which can be viewed as an *instance* from the olog in Fig. 8.

The design presented in Fig. 11 is generated form the previously presented designs by integrating the two ideas of central and distributed storage units.

The functor, as defined in 3, is a collection of mapping and transformations between two categories; in our case, from *Pro*-category to *Sol*-category. It contains all the mathematical and logical mapping and transformations that links the problem statements to candidate solutions.

In the case of the water system design mentioned in the previous section, the *Pro*-category represent the system main functional description. However, the *Sol*-category contains all possible designs and the mapping between the two categories are the functors; see Fig. 12.

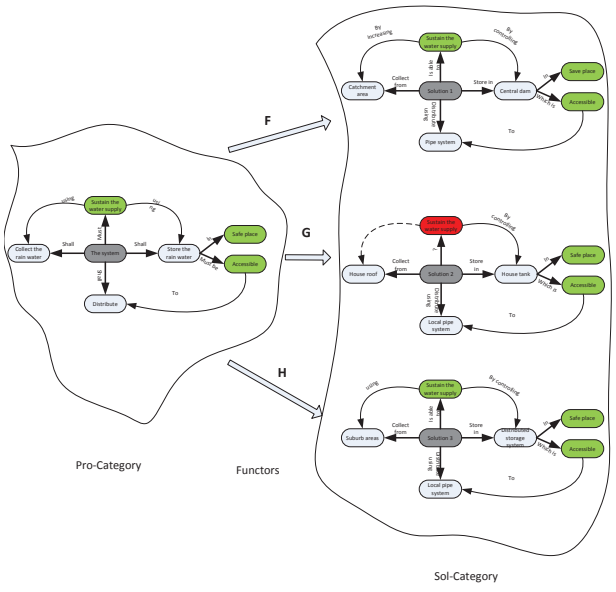


Fig. 12: Transformation or mapping in categorical language as a morphism (functor) between two categories; the problem-category (*Pro*) and the solution-category (*Sol*).

6 Conclusion

This paper introduced Category Theory as a formal foundation for model-based systems engineering, particularly due to its potential to formalize the transformation between design domains. The paper also presents a generalised view of the system as a category using Category Theory concepts where, objects of the category represent all the elements of the system and the arrows represent the relations between these components (objects). The relationship between these objects are the arrows or the morphisms in the category. In addition, the paper presented the *Olog* as a formal language for a given real-world situation discretion and requirement writing. The paper also showed how the olog can be used in model based system design using a water system design.

References

[1] M. M. Fokkinga, "A gentle introduction to category theory-the calculational approach," 1992.

[2] C. Hoare, "Notes on an approach to category theory for computer scientists," in *Constructive Methods in Computing Science*. Springer, 1989, pp. 245–305.

[3] D. S. Scott, "Relating theories of the lambda calculus," *To HB Curry: Essays on combinatory logic, lambda calculus and formalism*, pp. 403–450, 1980.

[4] D. E. Rydeheard and R. M. Burstall, *Computational category theory*. Prentice Hall Englewood Cliffs, 1988, vol. 152.

[5] B. C. Pierce, *Basic category theory for computer scientists*. MIT press, 1991.

[6] G. M. Reed, A. Roscoe, and R. F. Wachter, *Topology and category theory in computer science*. Oxford University Press, Inc., 1991.

[7] J. Herring, M. J. Egenhofer, and A. U. Frank, "Using category theory to model gis applications," in *4th International Symposium on Spatial Data Handling*, vol. 2, 1990, pp. 820–829.

[8] H. Rising III and A. Tabatabai, "Application of category theory and cognitive science to design of semantic descriptions for content data," Jan. 15 2008, uS Patent 7,319,951.

[9] K. Williamson, M. Healy, and R. Barker, "Industrial applications of software synthesis via category theorycase studies using specware," *Automated Software Engineering*, vol. 8, no. 1, pp. 7–30, 2001.

[10] J. L. Fiadeiro, *Categories for software engineering*. Springer Science & Business Media, 2005.

[11] S. Kovalyov, "Modeling aspects by category theory," *Proceedings of FOAL*, p. 63, 2010.

[12] Z. Diskin and T. Maibaum, "Category theory and model-driven engineering: from formal semantics to design patterns and beyond," *Model-Driven Engineering of Information Systems: Principles, Techniques, and Practice*, p. 173, 2014.

[13] H. Graves, "Category theory foundation for engineering modelling," draft 2013.

[14] D. Luzeaux, "A formal foundation of systems engineering," in *Complex Systems Design & Management*. Springer, 2015, pp. 133–148.
[15] ISO/IEC 15288, "Systems and software engineering system life cycle processes," 2008.
[16] M. Bunge, *Treatise on basic philosophy: Ontology II: A World of Systems*. Springer Science & Business Media, 1979, vol. 4.
[17] P. Micouin, *Model Based Systems Engineering: Fundamentals and Methods*. John Wiley & Sons, 2014.
[18] L. v. Bertalanffy, *General system theory: Foundations, development, applications*. Braziller. New York, 1968.
[19] C. Wasson, *System Analysis, Design, and Development: Concepts, Principles, and Practices*, ser. Wiley Series in Systems Engineering and Management. Wiley, 2006. [Online]. Available: http://books.google.com.au/books?id=KeJ3z_Yt1FEC
[20] A. L. Ramos, J. V. Ferreira, and J. Barceló, "Model-based systems engineering: An emerging approach for modern systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 1, pp. 101–111, 2012.
[21] R. Kalawsky, J. O'Brien, S. Chong, C. Wong, H. Jia, H. Pan, and P. Moore, "Bridging the gaps in a model-based system engineering workflow by encompassing hardware-in-the-loop simulation," *Systems Journal, IEEE*, vol. 7, no. 4, pp. 593–605, Dec 2013.
[22] J. Holt, S. Perry, R. Payne, J. Bryans, S. Hallerstede, and F. Hansen, "A model-based approach for requirements engineering for systems of systems," *Systems Journal, IEEE*, vol. 9, no. 1, pp. 252–262, March 2015.
[23] C. E. Dickerson and D. Mavris, "A brief history of models and model based systems engineering and the case for relational orientation," *Systems Journal, IEEE*, vol. 7, no. 4, pp. 581–592, 2013.
[24] A. Wymore, *Model-Based Systems Engineering*, ser. Systems Engineering. Taylor & Francis, 1993. [Online]. Available: http://books.google.com.au/books?id=CLgsYC3K2yAC
[25] C. Atkinson and T. Kuhne, "Model-driven development: a metamodeling foundation," *Software, IEEE*, vol. 20, no. 5, pp. 36–41, Sept 2003.
[26] C. Reviews, *e-Study Guide for: Knowledge Management In Theory and Practice: Psychology, Cognitive psychology*. Cram101, 2012. [Online]. Available: http://books.google.com.au/books?id=vD34WaHSd4kC
[27] D. I. Spivak, *Category theory for the sciences*. MIT Press, 2014.
[28] J. Adámek, H. Herrlich, and G. E. Strecker, "Abstract and concrete categories. the joy of cats," 2004.
[29] P. Vickers, J. Faith, and N. Rossiter, "Understanding visualization: A formal approach using category theory and semiotics," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 19, no. 6, pp. 1048–1061, 2013.
[30] D. I. Spivak and R. E. Kent, "Ologs: a categorical framework for knowledge representation," *PLoS One*, vol. 7, no. 1, p. e24274, 2012.

Mohamed Mabrok is currently working as a Postdoctoral fellow Robotics, Intelligent Systems and Control Lab at Computer, Electrical and Mathematical Science and Engineering, King Abdullah University of Science and Technology (KAUST). He is also lecturer at the Mathematics Department, Faculty of Science, Suez Canal University, Egypt. His main research includes Control Systems, Systems Engineering, System Design and Complex Systems. Dr. Mabrok received his B.Sc. in 2003 and M.Sc in 2009 in Applied Mathematics from the University of Suez Canal, Egypt. In 2013, he received his Ph.D. in Electrical Engineering from School of Engineering and Information Technology, University of New South Wales Canberra.

Michael J. Ryan received the B.E., M.Eng.Sc., and Ph.D. degrees in electrical engineering and the Graduate Diploma in management studies from the University of New South Wales, Canberra, Australia. He lectures and regularly consults in a range of subjects, including communications and information systems, systems engineering, requirements engineering, and project management. He is currently a Senior Lecturer with the School of Information Technology and Electrical Engineering, University of New South Wales, Canberra, Australian Defence Force Academy, Canberra. He is the author or co-author of 11 books, three book chapters, and over 100 technical papers and reports. Dr. Ryan is the Conference Chair of two international conferences each year. He is the Editor-in-Chief of the Journal of Battlefield Technology and the Chair of the Requirements Working Group in the International Council on Systems Engineering (INCOSE).