Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 44 (2015) 11 – 20

Procedia
Computer Science

2015 Conference on Systems Engineering Research

Measuring the complexity of a higraph-based system model: formalism and metrics

Hycham Aboutaleb[a]*, Bruno Monsuez[a]

[a]*Ensta ParisTech, 828 boulevard les Maréchaux, Palaiseau 91120, France*

Abstract

The exponential growing effort, cost and time investment of complex systems in modeling phase emphasize the need for a methodology, a framework and a environment to handle the system model complexity. For that, it is necessary to be able to measure the system model complexity. This paper highlights the requirements a model needs to fulfill to match human user expectations, presents a generic framework for designing complex systems, and suggests a graph-based formalism for modeling complex systems. Finally, a way to measure system model structural complexity based on Shannon theory of information is proposed and illustrated with an example.

Peer-review under responsibility of the Stevens Institute of Technology.

Keywords: Complexity; Metrics; Higraph; Information Theory; System Modeling .

1. Introduction

The concept of complexity is originated from systems theory. It could be defined as "a quality of an object with many interwoven elements, aspects, details, or attributes that make the whole object difficult to understand in a collective sense"[1]. Thus, complexity is an inherent property of systems.

Basic observations of complex systems tend to lead us to consider that the more elements they are composed of, the more complex they are. However, the quantity of elements composing a system is not the only factor. In fact, not only the quantity of elements is a factor, but also the quantity of interactions between these elements as well as the intensity of these interactions are to be taken into consideration. Moreover, the quantity of interactions with the system environment and their intensity are to be taken into account. Finally, another factor consists in the functions

* Corresponding author. Tel.: +33 6 67387190.
E-mail address: hycham.abou-taleb@ensta-paristech.fr

Peer-review under responsibility of the Stevens Institute of Technology.
doi:10.1016/j.procs.2015.03.057

to be delivered by a system and their status (from a failure point of view or safety point of view)[2]. To summarize, there are two main types of system complexity[34]:

- Internal complexity: It refers to the structural complexity of the system. It is a function of the quantity of elements, the quantity of interactions, and the intensity (or strength) of these interactions.
- External complexity: It refers to the system/environment interface complexity. It is a function of the quantity of interactions between the system and its environment, the intensity of these interactions, and the probability expected functions are performed.

This system *real* complexity is indeed reflected in the corresponding system model. In fact, the system *perceived* complexity is the model complexity.

While *real* complexity is an attribute of the system *itself*, the *perceived* complexity of the system is a relationship between an observer and the system observed: it has a relativistic aspect depending on the observer. It is an attribute of the *model* of the system.

Since System Engineering is nowadays usually model-based, the more complex a system model is, the more difficult and expensive is the design and the implementation effort. However, little literature can be found about system model and architecture complexity. This is mainly due to the fact that large complex systems development projects are not repeatable, making comparative studies hard to perform. Moreover, there is no widely used system model complexity measure.

In this paper is to introduce metrics and complexity measurements for system models, especially higraph-based models. Measuring complexity is useful to compare between system models. First, this paper defines the concept of complexity that is considered in this paper and highlights why measuring complexity is an industrial necessity. Then, a set of rules are identified for defining useful metrics. Another section summarizes the existing complexity measurements: an overview of the main complexity measurements is presented, including definitions and relevant properties. The next section defines metrics that are needed to evaluate a higraph-based model.

2. Reasons for measuring complexity

There are several intuitive reasons that make complexity measuring relevant and worthy:

- Cost:

Generally speaking, the more complex a system model, the more difficult it is to design, implement and use, and, intuitively, the more expensive it is. Although systems architecting phase of product development requires a small amount of the total development budget, deciding the architecture determines most of the total development cost. This is due to the fact that a late architecture modification architecture is very expensive. Therefore, it is prudent to try to avoid mistakes in systems modeling and architecting. Measuring complexity, and trying to reduce complexity, is one way of obtaining an optimal outcome: any unnecessary complexity is a risk for the final result and lowers the overall efficiency. Software business has for a long time tried to measure complexity of software. For example, a complexity measurement approach is proposed in [5] to predict the cost of software development projects with sufficient accuracy.

- System development management:

Given a measure of complexity, it is possible to identify the most complex subsystems. Intuitively, these subsystems shall require more resources and attention. Without this measure of complexity, the resources allocation might not reflect the distribution of complexity. Besides, empirical studies show that there is strong correlation between complexity and number of errors in a system development. But errors are not always due to complexity: measuring complexity of problematic subsystems gives an idea whether the problems are inherent in design or somewhere else. Measuring complexity of a ready product is useful by reducing its complexity in succeeding versions.

- Quality Assessment:

Since model-based design is used more and more in system engineering, and since it often follows an object-oriented approach, it is necessary to assess the merit of new system development technology. Measurements can only be interpreted when we also know the design. Criteria are defined to quantify factors such as reliability or usability. Each factor is composed of criteria such as traceability.

3. Measurement rules

Before defining complexity metrics, it is necessary to follow several rules to get relevant and useful indicators. These rules help designers to be consistent in their measurements, setting the framework for developing all kinds of measurement. Several rules shall be followed:

- Order:

If the measurement value of an element *A* greater than the measurement value of an element *B*, then the measurement value of the element *B* is less than the measurement value of an element *A*.

- Uniqueness:

The measurement value of an element *A* cannot be greater (or less) than itself: measuring the same property twice in same conditions shall give the same value.

- Numerical Value: The measurement value shall be a mapping from an observed relation system or element to a numerical relation system.

- Meaningfulness: The measurement value shall be understandable and its truth shall not depend on transformations on allowable scales, i.e. if the scale is changed the meaning shall be the same

Sometimes complex relationships between entities and properties lead to the definition of metrics that are then combined to get a high-level indicator.

4. Higraph-based modeling

4.1. Hierarchy issue

Graphs have been naturally used to represent and model problems since the emergence of computer science. Graph-based models give a visual and intuitive representation, as well as with required accuracy. They are a well-suited means to describe in a natural way all kind of systems, where nodes describe system entities and edges describe relations between them [6]. However when it comes to representing complex systems, the absence of hierarchy [7] is certainly one of the main defaults of graph-based representations.

To handle large amounts of data, it is often useful to have a classification or an order. One effective way to classify a set of elements is to use a hierarchical organization of this set of elements, introducing sometimes new order relations among the elements. With the hierarchy, in addition to be able to handle elements together, it becomes possible to handle subsets of elements together. There are two ways how to organize hierarchically a set: grouping and encapsulation.

- It is possible to group items based on similar properties or characteristics.
- It is possible to encapsulate many elements within a single element of a higher level and then consider only the properties of this element when an analysis is performed.

Consequently, we can indentify two types of models hierarchies. On one hand, there is the generalization, i.e.

hierarchy of types. The word type refers generally to a representation that gathers main properties of objects that have common characteristics [8]. One type allows to group elements with common characteristics. The mechanism of subtyping induces a hierarchy: an entity type T_2, derived from type T_1 has at least all the properties of an entity type T_1.
On the other hand, there is aggregation. The word aggregation refers generally to a representation that gathers elements into another higher-level element to hide them when necessary.

The higher-level element that encapsulates its contained elements has properties that are the emerging properties at this level due to the contained elements. Other names like nested hierarchy or container hierarchy are also common. Encapsulation decreases the complexity of the system model[9]. Finally, the hierarchy has an additional advantage: depending on the selected level, it is possible to observe different points of view.

4.2. Higraph

A higraph is a graph extended to include notions of depth and orthogonality and was introduced by Harel [10,11]. In other words:

Higraph = Graph + Depth + Orthogonality

Definition (Higraph).
A higraph is a quadruble $H = (B; E; \rho; \Pi)$ where :
- *B* is the set of blobs (or nodes);
- *E* is the set of edges.
- *ρ* is the hierarchy function. It assigns to each blob $b \in B$ its set of sub-blobs *ρ (b).*
- *Π* is the orthogonality (or partitioning function) defined as $\Pi : B \rightarrow 2^{B \times B}$, associating with each blob $b \in B$ some equivalence relation *Π(b)* on the set of sub-blobs, *ρ (b)*.

By its definition, the depth, shown by a higraph is defined by the enclosure of one node within another.

5. Metrics

5.1. Direct metrics

Since complexity needs an unambiguous framework to be defined clearly and to be measured relevantly, basic metrics are identified first. In that purpose, a set of properties are identified as useful for the calculation of complexity of a higraph-based model.

- Size:

The most obvious and useful attributes of a model is its size, which can be measured statically for static as well as dynamic models. The most intuitive way is to take into account the number of nodes and the number of edges.

- Depth:

The depth of a higraph-based model is the highest number of levels between the top node and the lowest level node.

- Width:

The width of a higraph-based model is the highest number of nodes at any one level.

5.2. Indirect metrics

- Density:

It measures the node constituents to the number of nested components. This metric is used to identify the density of nested elements.[12,13]

- Type Variety

The number of types in a set of elements is a good indicator of variety if all the types are of equal importance, which is usually not the case[14].
This index is suitable since it possesses the following properties:
  - For symmetric element types it equals the number of element types.
  - The introduction or disappearance of a marginal type does not cause a discrete change in the variety index.

- Interface Load

This index measures the average number of interface inputs into an element and the average number of interface outputs of an element and provides an overall measure for the whole model[15].

5.3. Shannon's entropy

Statistical theory of information, as developed by Shannon[16], is an answer to the question: given a set of messages $\boldsymbol{m_i}$ each of which occurs with probability $\boldsymbol{p_i}$, what is the amount of information they convey. The first step is to determine the amount of information provided by a single message $\boldsymbol{m_i}$, which is:

$$I(m_i) = -\log_2 p_i \tag{1}$$

Definition (Shannon's entropy). Let then X be a set of discrete random variables with values $x_1; x_2; ...; x_n$ with x_i having probability $\mathrm{p_i}; (1 < \mathrm{i} < \mathrm{n})$ Shannon's entropy H is defined as:

$$H(X) = -\sum_{1}^{n} p_i \log_2 p_i \tag{2}$$

Consider a set S containing N_S states. We can split S into k independent subsets such that (Figure 1):

$$S = \bigcup_{i=1}^{k} S_i, S_i \neq 0, \forall i \tag{3}$$

And

$$N_S = \sum_{i=1}^{k} N_{S_i} \tag{4}$$

The probability of a state x belonging to S_i is:

$$p_i = -\sum_{i=1}^{k} p(x \in S_i) = N_{S_i} / N_S \tag{5}$$

The complexity of this system is thus:

$$H(S) = -\sum_{i=1}^{k} p_i \log_2 p_i = -\sum_{i=1}^{k} N_{S_i} / N_S \log_2 (N_{S_i} / N_S) \tag{6}$$

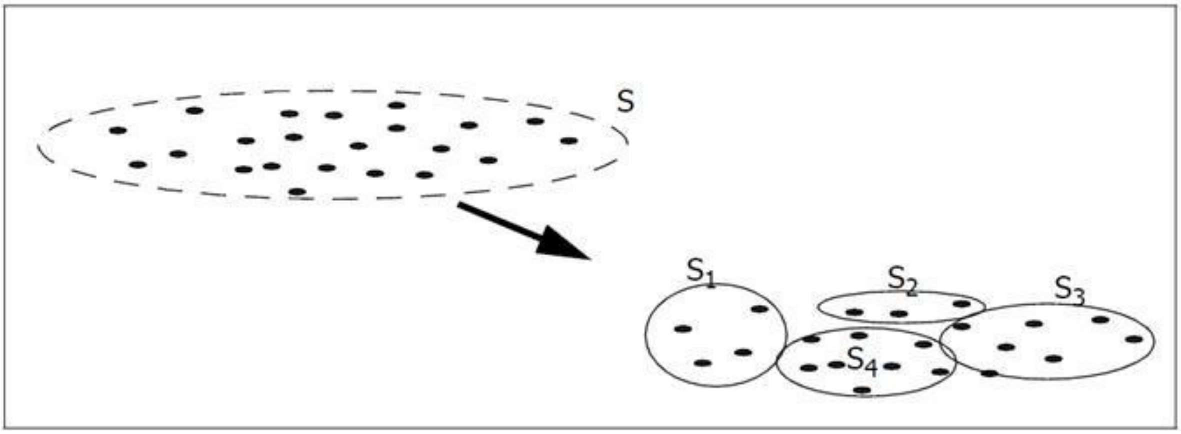

Figure 1 Decomposing a set

By changing the perspective from working with a large set S of N_S individual states (x) to a collection of subsets containing a smaller number N_{S_i} of states ($x \in S_i$), the whole set complexity has been replaced with the probability weighted sum of the complexity found within each subset. This is a very powerful principle in design: a complex problem is decomposed into a set of smaller problems with smaller complexity. Besides, the global complexity is the same.

6. Higraph-Based System Model Structural Complexity

A higraph model M entropy intuitively depends on the number of blobs, the number of edges, the hierarchy and the orthogonality [17].
We use Shannon's entropy as an indicator of the complexity.
We get the entropy of the model higraph as follows:

$$H = H_B + H_E + H_\rho + H_\Pi \tag{7}$$

To evaluate the complexity of a higraph M , it is consequently necessary to get the complexity get each term separately.

- H_B :

$$H_B = H(B) = -\log_2(1/|B|) = \log_2(|B|) \tag{8}$$

- H_E :

$$H_E = H(E) = -2\log_2(1/|E|) = 2\log_2(|E|) \quad (9)$$

It takes into account the head and the tail of the edge.

- H_ρ:

H_ρ relates to the number of hierarchical relationships between the elements of the model N. Multiple locations of an element, i.e. an element has several parents, are taken into account.

$$N = \sum_{x \in M} |\rho(x)| \quad (10)$$

It is obvious that if there is no hierarchy, $N = |B|$, i.e. the diagram contains all the elements at the same level.

$$H_\rho = -2\log_2(1/|N|) = 2\log_2(|N|) = 2\log_2(\sum_{x \in M} |\rho(x)|) \quad (11)$$

Where we take into account parent and child relationship.

- H_Π:

$H_\Pi = H(M_\Pi)$, where M_Π is the Type Higraph associated to the higraph M.
Let M_Π be a Type Matrix higraph.
Let M be a Model higraph.
Let $g : M \to M_\Pi$ a morphism that associates to each element (object, flow, attribute) *x* of the Model higraph M to its type, with M_Π, the Model Type Higraph.

We have:

- $\forall x \in M, g(x) \in M_\Pi;$
- $\forall x \in M, g(\rho(x)) \subset \rho(g(x));$
- $\forall t \in M_\Pi, g(\Pi_t(x) \subset \rho(t).$

Besides, $H_\Pi = (B_{M_\Pi}; E_{M_\Pi}; \rho; \Pi)$ have the following properties:

- $E_{M_\Pi} = 0$, i.e. there is no edge;
- $\forall x \in B_{M_\Pi}, \Pi(x) = \rho(x)$; i.e. all elements are of the same type.

We have:

$$H_\Pi = H(B_{M_\Pi}) + H(E_{M_\Pi}) + H_\rho(M_\Pi) + H_\Pi(M_\Pi),$$

where:

- $H(B_{M_\Pi}) = \log_2 |B_{M_\Pi}|$
- $H(E_{M_\Pi}) = 0$

- $H_\rho(M_\Pi) = 2\log_2(\sum_{x \in M_\Pi} |\rho(x)|)$
- $H_\Pi(M_\Pi) = 0$

Thus, we get the entropy of the model higraph as follows:

$$H = H_B + H_E + H_\rho + H_\Pi$$

i.e.

$$H = \log_2|B| + 2\log_2|E| + 2\log_2(\sum_{x \in B} |\rho(x)|) + \log_2|B_\Pi| + 2\log_2(\sum_{x \in M_\Pi} |\rho(x)|) \tag{12}$$

7. Example

We take the following example: an electric toothbrush. A detailed view of the attributes and functions for the hardware and software is shown (Figure 2). An electric toothbrush is of type *System*. It can contain elements of types *Hardware* or *Software*. *Hardware* may then contain elements of types *Functions*, *Physical Attributes* and *Logical Attributes*. *Software* may contain elements of types Attributes and Functions. Elements of different types are separated by dotted lines.

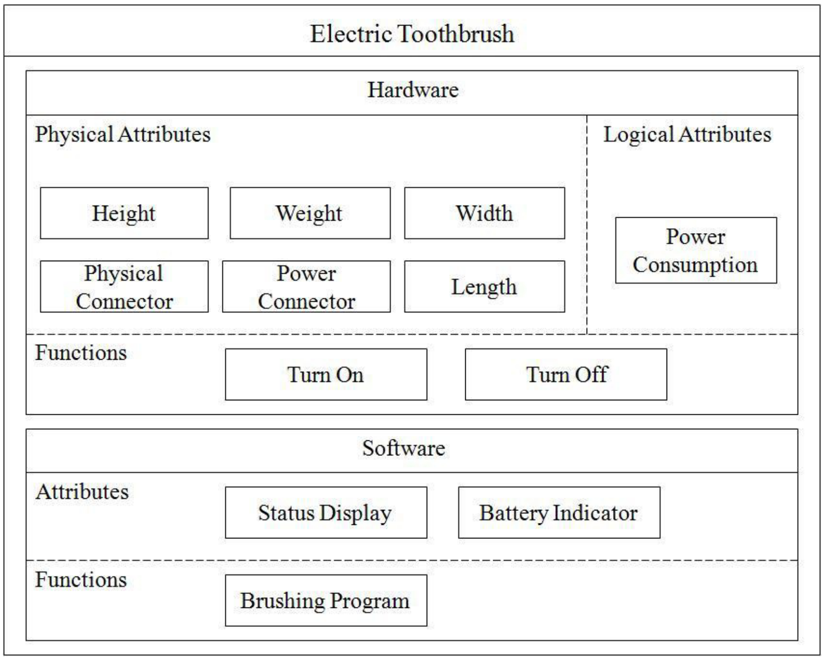

Figure 2 Detailed system breakdown for an Electric Toothbrush

The entropy of the type higraph M_Π is:

$$H(M_{\Pi}) = \log_2 |M_{\Pi}| + 2\log_2 (\sum_{x \in M_{\Pi}} |\rho(x)|)$$

$$H(M_{\Pi}) = \log_2 (7) + 2\log_2 (5)$$

The entropy H of the Model M is:

$$H(M) = H_B + H_E + H_\rho + H_\Pi$$, where:

- H_B:

Elements are: *Hardware, Software, Height, Length, Width, Weight, Power Consumption, Physical Connector, Power Connector, Turn On, Turn Off, Battery Indicator, Status Display, Brushing Program*.

$$H_B = \log_2 (14)$$

- H_E:

There are no edges in the higraph. Therefore, we have:

$$H_E = 0$$

- H_ρ:

$$H_\rho (M) = 2\log_2 (\sum_{x \in M} |\rho(x)|) = 2\log_2 (12)$$

- H_Π:

$$H_\Pi = \log_2 (7) + 2\log_2 (5)$$

8. Conclusion

Shannon's information entropy can be used as an indicator of complexity. Its value depends on the amount of details, elements and relationships between them, as well as the number of hierarchy levels. As shown, according to Shannon's information entropy, smaller sets mean less complexity. The choice of aggregation allows dealing with subsets separately to handle this complexity. Moreover, the complexity measurement is a relevant metric to compare different architectures for the same system.

Handling complexity has impacts on the design whether it is cost, effort, planning or safety. However, the fact that there is little literature is mainly due to the fact that system architects use their intuition to measure and handle complexity. For models that are easily glanced at, they are able to measure if they are excessively complex or not. For large models, complexity measure is useful to identify the most complex subsystems, since they are the ones that need the most attention due to their expected impact on the overall design. Many studies show that less complex systems are more likely to be more successful from a business point of view.

References

1. El-Haik B, Yang K. The measurement of a design structural and functional complexity. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(4):527535, July 1998.
2. Braha D. The components of complexity in engineering design. *IIE Transactions*, 31(10):925934, October 1999.
3. Martin JN. *Systems Engineering Guidebook: A Process for Developing Systems and Products*. CRC Press Inc, 1996.
4. Kinnunen MJ. *Complexity measures for system architecture models*. Master of science in engineering and management, Massachusetts Institute of Technology, 2006.
5. McConnell S. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, 1996.
6. Lambers L. *Certifying Rule-Based Models using Graph Transformation,* PhD thesis, Technical University of Berlin, Germany, 2009.
7. Drewes F, Hoffman B, Plump D. *Hierarchical graph transformation*. Journal of Computer and System Sciences, 64(2):449–283,2002.
8. CardelliL, Wegner P. *On understanding types, data abstraction, and polymorphism*. ACM Comput. Surv., 17(4):471–522, 1985.
9. Ahl V, Allen TFH. *Hierarchy Theory - A Vision, Vocabulary, and Epistemology*. Columbia University Press, 1996.
10. Harel D. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(5):231–274, 1987.
11. Harel D. On visual formalisms. *Communications of the ACM*, 31(5):514–530, 1988
12. Kumar S, Tomar P, Nagar R, Yadav S. Coupling metric to measure the complexity of component based software through interfaces. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(4):157162, 2014.
13. Narasimhan VL, Hendradjaya B. Some theoretical considerations for a suite of metrics for the integration of software components. *Journal Information Sciences*, 177(3):844864, 2007.
14. Straathof. SM Shannon's entropy as an index of product variety. *Economic Letters*, 94(2):297303, 2007.
15. Kumari U, Upadhyaya S. An interface complexity measure for component-based software systems. *International Journal of Computer Applications*, 36(1):4652, 2011.
16. Shannon CE. A mathematical theory of communication. *Bell SystemTechnical Journal*, 64:379–423,623–656, 1948.
17. Rossi M, Brinkkemper S. Complexity metrics for systems development methods and techniques. *Information Systems*, 21(2):209–227,1996