Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2012) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

New Challenges in Systems Engineering and Architecting
Conference on Systems Engineering Research (CSER)
2012 – St. Louis, MO
Cihan H. Dagli, Editor in Chief
Organized by Missouri University of Science and Technology

An Empirical Investigation of System Changes to Frame Links between Design Decisions and Ilities

J. Clark Beesemyer, Adam M. Ross, and Donna H. Rhodes

Massachusetts Institute of Technology, Systems Engineering Advancement Research Initiative
Building E38-576, 77 Massachusetts Avenue
Cambridge, MA 02139 USA

Abstract

Maintaining system performance in the presence of uncertainties in design and operating environments is both challenging and increasingly essential as system lifetimes grow longer. In response to perturbations brought on by these uncertainties, such as disturbances, context shifts, and shifting stakeholder needs, systems can continue to deliver value by being either robust or changeable. These lifecycle properties, sometimes called "ilities", have been proposed as means to achieve system value sustainment in spite of changes in contexts or needs. Intentionally designing for these lifecycle properties is an active area of research, and no consensus has formed regarding how these and other "ilities" might trade off. This paper describes ongoing research that investigates empirical examples of system changes in order to characterize these changes and to develop a categorization scheme for framing and clarifying design approaches for proactively creating ilities in a system. Example categories from the data for system changes include: the perturbation trigger for the change, the type of agent executing the system change, and the valid lifecycle phase for execution. In providing a structured means to identify system change characteristics, this paper informs future research by framing possible relationships between ilities and design choices that enable them.

Keywords: Changeability; Evolvability; Adaptability; Flexibility; Change Mechanism, Categorical Cluster Analysis

1. Motivation

The beginning phases of system development and conceptual design require careful consideration, as these decisions will have significant influence on system lifetime performance and are usually made with incomplete system knowledge. Decision makers may improve their capacity to discriminate between system concepts and design choices by measuring a system's "ilities" such as changeability, scalability, and survivability. These ilities may enable systems to respond to shifts in contexts and needs in order to ensure system functionality and adequate performance over time. A system may be designed to change in response, or remain robust to perturbations in order to avoid deficiencies or failures. Characterizing

system changes through empirical examples may inform research on how system ilities relate to each other across various system and domain types. This research attempts to analyze mechanisms that allow system changes to occur, and propose a framework for allowing system designers to map vague, yet desirable, ilities to prescriptive system design principles.

2. Background Information

In order to explore the impact early design decisions may have on lifecycle properties, called ilities, it is necessary take a broad perspective, looking at the environment in which a system operates. This environment, or operational context of the system, when combined with a set of stakeholder needs, is called an "epoch" and characterizes the key exogenous factors that impact the ultimate success of a system. Since the goal of any system is to meet these needs in various contexts, delivering benefit at cost, or value, across changing epochs is a measure of success as defined by individual stakeholders of the system [1].

2.1. Value robustness

System designers are required to make design choices early in system lifecycles that impact operational performance in alternative future contexts. A value robust system is one where the system maintains value delivery in spite of changes in needs or contexts (epochs). While value robust systems may cost more upfront, the added security of delivering value in uncertain epochs may be worth the extra costs to system stakeholders and decision makers.

Value robustness can be evaluated over a system lifespan, or the "cradle-to-grave" period of time for a system. In this research, this lifespan is known as an *era*. Eras are ordered sequences of *epochs*, periods of time where the system experiences fixed context and value expectations (needs). Further characteristics of an epoch include static constraints, available design concepts, available technology, and articulated attributes [1]. If a possible system design cannot maintain expected value levels in one or more possible epochs, that design may be deemed unsatisfactory by the stakeholders of the system. A system that meets or exceeds value expectations in all possible epochs represents the most desirable value robust system.

Transcending traditional systems engineering practices of optimizing to specific contexts and needs, epoch-era analysis methods and designing for value robustness enrich the understanding of the temporal aspect of system lifecycles [1].

There are various means to achieve value robustness, and that is where the lifecycle properties known as ilities come into play. A system may achieve value across epochs by passive or active means. For example, a system that requires thermal control could use shielding or special materials to passively control thermal states, remaining robust to changing contexts. It may also use maneuverability or cooling systems to actively control operating temperatures.

A system may respond to shifting epochs and decreased performance by changing in either form, function, or operations. Systems engineers make use of ilities in an effort to capture different ways a system can perform. Designing for these ilities, however, presents a problem to decision makers in that they are in many cases ill-defined, fuzzy descriptions that are hard to measure, verify and validate. Because of this, systems can get away with advertising various ilities without substantiating evidence proving or disproving these alleged system properties.

2.2. Ilities as outcomes

The goal of designing for ilities is to improve how a system responds to perturbations based on various measures during different lifecycle stages. Ilities offer decision makers means of discriminating between different design concepts and systems for fulfilling the needs that are required. These system properties

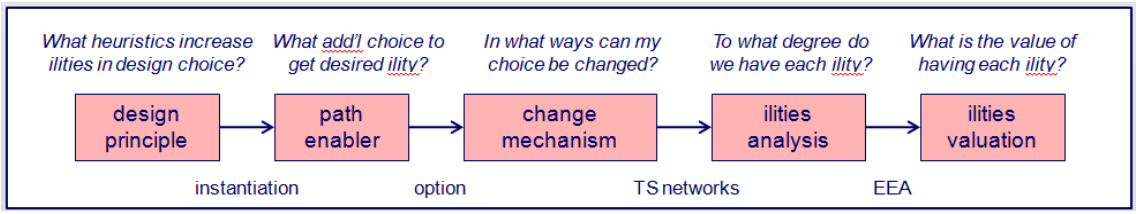

Figure 1. Relationship between Design Principles and Ilities [3]

are used as labels to describe "the ability" to make certain changes to your initial design (or resist changes). For example, Ref [2][1] describes system ilities, such as flexibility, adaptability, scalability, and modifiability, which enable a system to remain valuable through lifecycle changes in function, form, or operations.

The problem with ilities is that without broadly agreed upon definitions and metrics, they can be ambiguously used to describe system properties. As ilities generally refer to positive system properties, system designers as well as decision makers naturally desire their systems to contain many of these properties without necessarily recognizing the consequences of implementation in design. This research aims to find a better means of determining which ilities are present in different system changes and map those ilities to various design principles. To better understand this, a concept of how ilities may be mapped back to their respective design principles is shown in Figure 1[3].

When stakeholders identify an ility as a desired property of a design, the ultimate goal of added value to the system from having this ility represents one end of this relationship (Figure 1). As an example, a stakeholder in a wildfire tracking satellite system, such as the one described in Ref [4] called FireSat, may desire agility in the system to respond to new fires rapidly. To evaluate different system concepts, decision makers must have a way to measure the degree to which each alternative exhibits the desired ilities. However, the language used to describe these system properties lacks common structure in modern systems engineering, particularly more relative ilities, such as agility. For this reason it is imperative that decision makers clearly define what each ility means to them, and how it may be measured, and therefore able to be verified, in the system. In agility for example, relativity in response time requires decision makers to propose a baseline execution time for comparison. FireSat decision makers may require specific slew rates or inclination change rates for example. What may be an agile system to one person in one context may not be for another. This specificity is an important component to ility-based analysis.

3. Change Options

A change mechanism is a description of the transition of a system from one state to another, or the path [1],[5]. Each change mechanism is made possible by one or more path enablers, which provides a change option to execute the change mechanism. In FireSat, a change option could be burning a thruster to change orbit or using momentum wheels to change spacecraft attitude. A change option may be broken into numerous components that can accurately describe how each mechanism is executed and what costs are associated with that option as shown in Figure 2 [3].

A change option starts with a path enabler that gives the opportunity to execute some change. In FireSat, path enablers could be thrusters or internal momentum wheels. Path enablers come with initial implementation costs, as well as start dates and possibly expiration dates (momentum wheels may be restricted from use in early commissioning phases). The specific mechanism will allow for one or more optional new end-states (a new orbit or new attitude for example). There may be certain states that are unachievable in certain epochs (an orbit deemed too dangerous for levels of radiation or debris). There may be some carrying cost as well for a certain change mechanism. FireSat has the carrying cost of power to keep momentum wheels spinning while in orbit. Then, if execution of the mechanism is desired, there may be an execution cost associated with that change (power, time, etc.).

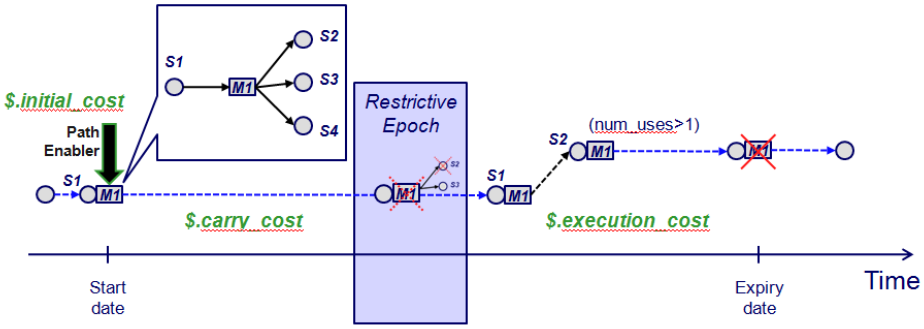

Figure 2. The Change Option (made up of the path enabler and change mechanism) [3]

The execution itself may be characterized by which system end state is desired and in which epoch the mechanism is executed. The execution will have an initialization time and a duration, as well as certain prerequisites. For example, to operate FireSat's thrusters to change orbit, the satellite must contain the needed fuel, be controlled by a ground-station, and have set start and duration times for firing with a possible required "warming-up" or initialization time. The overall system change may be characterized by one or multiple ilities (possibly flexibility, agility and scalability in the FireSat example). Mechanisms may be reusable, like a liquid thruster may be, or one-time use only like solid rocket motors. The existence of many available system changes and their associated "ility" characterizations may be used to determine whether a given system has particular system properties [3].

Overall, a specific change mechanism can be traced back to the component that enabled it, the path enabler. Path enablers may enable one or multiple change mechanisms. For example, the thrusters could enable orbit changes as well as attitude changes in the FireSat system. The ultimate goal of these relationships however, is not to define the specific path enabler of the system, but to obtain a more meaningful principle in design. A design principle is a guiding thought based on empirical deduction of observed behavior or practices that proves to be true under most conditions over time [6]. Narrow research on specific path enablers of a specific system, while useful for particular system insights, may lead to dissimilar results in different systems or domains. Tracing back to more general design principles may be more universal and useful to systems engineering as a whole.

3.1. Characterizing change mechanisms

In an effort to explore the many different change mechanisms present in various historical and modern-day systems, a method of defining these mechanisms was created. The method used to characterize different changes uses a generic change statement to enable a classification process for different changes implemented in systems. This statement will become the basis for mapping a change to its representative ility or ilities according to its characterization. The generalized structure of this change statement is:

In response to "cause" in "context", desire "agent" to make some "change" in "system" that is "valuable".

The components in quotations are defined for each specific system change evaluated. This classification is used as the basis for mapping different change types to various ilities as shown in the draft framework in Figure 3. The first two components deal with the cause and context of the change. This refers to the reason the change is occurring in response to what type of perturbation exposed to the system and if the perturbation is conditional or general. The "system" is then defined by the abstraction being changed; is it the architecture changing, the design, or a specific system already in use? The aspect of the system (form, function or operations) as well as the lifecycle phase is described as well. For each change mechanism evaluated, there will be a defined change agent, or whether the initiating force is internal or external to the defined system boundaries. The change itself is then defined by type and effect,

"cause & context"		"system"			"agent"	"Δ" (in ________)				"valuable"			
Why cause	Where context	What entity	What aspect	When phase	Who agent	What parameter change type	What effect (scale) (or)	What effect (amt)	What potential states	When timing	When span	For what resources	For what benefit
perturbation	specificity	abstraction	aspect	LC phase	executes	param. type	level	set	target range	reaction	duration	cost	utility
finite	circumstantial	architecture	form	pre-ops	internal	level	bigger	more	one	sooner	shorter	cheaper	more
shift	general	design	function	ops	external	set	smaller	less	few	later	longer	expensiver	less
none	any	system	operations	inter-LC	none	any	not-same	not-same	many	always	same	same	same
any		any	any	any	any		same	same	any	any	any	any	any
							any	any					

--	--	--	function/ops	ops	--	set	not-same	many	**versatility**
shift	circumstantial	system	--	ops	--	--	same	--	**robustness**
shift	circumstantial	system	form	ops	none	level	same	few	**classical passive robustness**
shift	general	architecture	--	inter-LC	--	--	--	--	**evolvability**
--	--	--	--	--	internal	--	--	--	**adaptability**
--	--	--	--	--	external	--	--	--	**flexibility**
--	--	--	--	--	--	level	not-same	--	**scalability**
--	--	--	--	--	--	set	not-same	--	**modifiability**
finite	circumstantial	--	--	ops	--	--	--	--	**survivability**
--	--	--	form	ops	--	--	--	--	**reconfigurability**
--	--	--	--	--	--	--	not-same	--	**agility (timing=sooner or span=shorter)**
--	--	--	--	ops	--	set	more	--	**extensibility**

Figure 3. Mapping Change Mechanisms to Ilities

whether the change is to a level or set of variables, and how much or many of that change is present. The number of potential end-states for each change mechanism is also captured. Since value-based analysis is useful to decision makers, change mechanisms may be even more deeply defined by individual preferences in the value they deliver. This value requires a baseline system or performance level for comparison, as value is usually characterized by aspects such as faster, shorter, cheaper, or other similar comparative ideas.

Based on the responses to the above components of change mechanisms, various ilities may be observed as seen in a few examples at the bottom of Figure 3. A change mechanism may be defined as flexible if an agent executes the change externally from the system, but that does not mean that change mechanism cannot also be labeled by reconfigurability, scalability, versatility, or other such ilities.

This represents a first step for calling out distinct "bases" (or "categories") for characterizing ilities. It is expected that the framework will demonstrate that various subtypes of particular ilities exist and may help to make these more explicit. For example, robustness is a "superset" ility according to the framework since it is characterized by several dimensions left as "any", meaning it includes any choice of those dimensions as subsets of the concept. Particular subtypes of robustness, such as "classical passive robustness" as listed in the figure, are specified by selecting particular choices in the "any" columns of the general robustness concept. In this way superset-subset relationships can be identified (through counting the number of "any" values in the columns, one can identify the "higher level" ilities).

This framework requires a clear description of the change mechanism under evaluation and the system parameter being specified. One can take the exact same change mechanism and redefine the system boundaries, or parameter which is altered, and come up with very different corresponding ilities. Even with very well defined ilities, altering the subjectively defined scope of analysis (e.g., system boundary) makes it possible to portray a system as more flexible or adaptable, or more modifiable or reconfigurable.

3.2. Levels of analysis

There are various levels or "abstractions" in which analysis of a system can be performed. Every system has a corresponding architecture, even if not explicitly defined, which is made up of the blueprint from which all designs of the system originate. The architecture may contain one or many different designs of the system. From the design level, a specific system may be constructed and implemented. Therefore, there are many instances of designs from a given architecture, and many instances of systems

from a design. To clarify, consider the Apple iPhone. The iPhone has an overall architecture, one being the iPhone 4 architecture. Within that architecture there are different instances of designs, for example there are 16 GB and 32 GB designs, and different cell-phone carrier designs, all considered variants of "iPhone 4". Within each of those designs there are different instances of systems, for example my AT&T iPhone 4 32GB vs. your AT&T iPhone 4 32GB.

These distinctions become relevant when analyzing the evolvability of a system. By definition, evolution of a system must take place between generations of a system [7]. For the purposes of this research, a new generation of a system occurs when changes are made to the system architecture. Pliability is described as the ranges of parameters within a system architecture that yield viable system designs [8]. If a data storage device is offered as giving 100 to 500 gigabytes of storage, a terabyte of storage would be considered outside the pliable range of that architecture. A designer could create a terabyte storage device, but it would require a change in architecture and loss in guarantee that the prior architecture would remain viable. These breaks in architecture are where changes in generations occur and are exemplified by the iPhone 3GS vs. the iPhone 4.

The mapping framework captures the difference in the level of analysis being executed in the abstraction of the change. Important to note is even if a specific system is scalable in design, it does not necessarily mean it is scalable in specific design instances. For example, designs of a rocket family may be scalable in deliverable mass to orbit, but a specific instance of the design being built may no longer be scalable. For instance, the architecture of Atlas V rockets allows for scalable designs with 1 or 3 common booster cores [9]. However, once a design is chosen, it ceases to be scalable in the same way. A single common booster core rocket under construction cannot be scaled to a 3 common booster core rocket.

4. Empirical Cases

The approach in this research involves deriving design principles and heuristics from empirical observations of historical and present-day systems. Refinement of the framework to identify different cases will allow for a structured means of analyzing ilities, change mechanisms, path enablers and their corresponding design principles in many different domains and systems.

4.1. Change database

A database to hold the data for different change mechanisms in various systems experiencing a wide variety of changes was created to aid in research analysis of connections between mechanisms and ilities and any trends therein. This database was created with the change mechanism framework in mind and attempts to capture sufficient data about actual system changes in a structured manner. The database gathers information and justifications for each of the components discussed above, in the change mechanism framework section of the paper. The database also provides a means to gather other available information including cost, cost context, preliminary information, product details, and ility mapping. Depending on the change details, various ility types are highlighted to signify a specific ility-set that may be attributed to that change. Informed by Ref [1], these ilities and their conditions are being constantly refined and augmented to apply to any generic change for any system.

4.2. Preliminary insights

Preliminary research shows that precise language in system abstractions and aspects is critical in evaluating how systems change from one state to another. A system may use the same path enabler to get to many different states, for many different reasons, in response to many different perturbations. Depending on which parameter is specified, or the defined system boundaries, or other similar distinctions, many different ilities may be shown as present in a single change mechanism, let alone

system. For example, take the F-14 Tomcat variable wing sweep aircraft. One particular change mechanism refers to the F-14's central air data computer to control the wing sweep angle in order to maintain high lift-to-drag ratios in different flight regimes. Because this change mechanism is initiated from an internal computer (agent) it is considered an adaptable change. The parameter specified is the lift-to-drag. In response to a shift in environment, the aircraft changes form to maintain the parameter. Therefore the database yields it as a changeable, adaptable, reconfigurable, and robust change. If, however, the same exact change mechanism is evaluated with a different parameter, like wing sweep angle, the change is no longer robustness (maintaining lift-to-drag ratio), it is scalable (change the level of the wing sweep angle).

This at first makes the framework seem futile, if, depending on semantics, various ilities may be attributed. For this reason it may be invalid to state that hinged wings on aircraft, as a path enabler, relates directly to robustness in a design. However, this research shows that clarity in ility and change statements is crucial if a designer or decision maker is going to begin to discriminate between designs. It is no longer acceptable to call a design simply "robust," or "flexible." Those characteristics are meaningless unless accompanied by corresponding information as discussed in the change mechanism section of this paper. It is common in literature for the A-10 to be referred to as a robust design or a survivable design [10],[11]. This framework shows it is more useful to claim system properties in a more complete way that can be measured and verified by decision makers and stakeholders. For example, one may say that the A-10 is adaptively survivable and reconfigurable to finite disturbances of small arms fire during close air support when its fuel tanks respond to bullet holes by self-sealing in combat. This statement is not only more specific and informative to stakeholders, but it also gives systems designers something to test to and verify (e.g., must self-seal bullet holes to a specific caliber in a specific time period).

Additionally, stakeholder value is subjective, and clear representation of what stakeholders actually desire in their system may be difficult to obtain. Imprecise use of ilities can exacerbate these miscommunications. When stakeholders and designers agree on the precise need ("a flexible system" for example) the likelihood of satisfying stakeholders is significantly increased.

The multitude of ilities present in various changes also demonstrates that ilities are not mutually exclusive. There are interrelations between the ilities, as well as subset-superset relationships, on different levels of analysis. This research is in its infancy, but aims to explore more of these relationships as the change database grows to cover more systems and more change mechanisms.

Various path enablers may enable one or multiple different change mechanisms, all with differing ility types. If there are commonalities or trends in these path enablers, they may lead to design principles that are successful in a wide range of system domains.

4.3. Categorical Cluster Analysis

In order to explore the database and find relationships between change options, a cluster analysis was applied to the database. Since this data is categorical, not numerical, standard methods of clustering, such as k-means, are difficult to apply. Instead, a model-based clustering algorithm, COOLCAT, was used [13], [14]. COOLCAT determines clusters by minimizing the expected entropy in a specified number of clusters. Entropy may be seen as a measure of similarity between records; the more similar records are, the lower the entropy of the cluster. Preliminary results indicate there are distinct clusters that this algorithm finds in the database. Current research is exploring differences between these clusters and attempting to find correlations with existing ilities as well as undefined candidate ilities.

5. Conclusion

Maintaining system performance in the presence of uncertainties in design and operating environments is both challenging and increasingly essential as system lifecycles grow longer. Designing robust or

changeable systems in response to such perturbations may cost more up front and seem less beneficial to early on. Despite this desire to implement ilities in designs, a clear method is still lacking for identifying "good" designs with respect to preferred ilities. Further, no consensus in the systems engineering community has been reached in how these ilities may trade-off. This paper describes ongoing research that investigates empirical examples of system changes in order to characterize these changes and to develop a categorization scheme for framing and clarifying design approaches for proactively creating ilities in a system. In providing a structured means to identify system change characteristics, this paper informs future research by framing possible relationships between ilities and design choices that enable them. Classifying change mechanisms, alone, has demonstrated the importance of having such a structured means of evaluation for decision makers. Stakeholders may benefit from a method to properly communicate ility preferences to designers. Since ilities are not inherent properties of a system, but more outcomes of design principles that enable change options, designers may have a means of identifying successful design heuristics in pursuit of providing acceptable design concepts for stakeholders.

Acknowledgement

The authors gratefully acknowledge funding for this research provided through MIT Systems Engineering Advancement Research Initiative (SEAri, http://seari.mit.edu) and its sponsors. Assistance in categorical clustering and data mining given from Nirav Shah, MIT SEAri Doctoral Candidate.

References

[1] Ross, A. And Rhodes, D. "Using Natural Value-Centric Time Scales for Conceptualizing System Timelines through Epoch-Era Analysis." *INCOSE International Symposium 2008*, Utrecht, the Netherlands, June 2008.

[2] Ross, A.M., Rhodes, D.H., and Hastings, D.E., "Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining Lifecycle Value," *Systems Engineering*, Vol. 11, No. 3, pp. 246-262, Fall 2008.

[3] Ross, A.M., "Anatomy of a Change Mechanism," SEAri Working Paper WP-2011-1-1, 2011, http://seari.mit.edu/papers.php [cited 02 Sep 2011].

[4] Wertz, James Richard. and Larson, Wiley J. *Space mission analysis and* design / edited by James R. Wertz and Wiley J. Larson Kluwer Academic, Dordrecht ; Boston : 1991

[5] Ross, A.M., *Managing Unarticulated Value: Changeability in Multi-Attribute Tradespace Exploration*, Doctor of Philosophy Dissertation, Engineering Systems Division, MIT, June 2006.

[6] Wasson, Charles S., Systems Analysis, Design, and Development: concepts, principles, and practices, Hoboken, NJ: John Wiley & Sons, 2006

[7] Beesemyer, J.C., Fulcoly, D.O., Ross, A.M., and Rhodes, D.H., "Developing Methods to Design for Evolvability: Research Approach and Preliminary Design Principles," 9th Conference on Systems Engineering Research, Los Angeles, CA, April 2011.

[8] Mekdeci, B., Ross, A.M., Rhodes, D.H., and Hastings, D.E., "System Architecture Pliability and Trading Operations in Tradespace Exploration," 5th Annual IEEE Systems Conference, Montreal, Canada, April 2011.

[9] Alliance UL. *Atlas V Launch Services User's Guide*; 2010.

[10] Richards, M.G., Ross, A.M., Hastings, D.E., and Rhodes, D.H., "Two Empirical Tests of Design Principles for Survivable System Architecture," INCOSE International Symposium 2008, Utrecht, the Netherlands, June 2008.

[11] Richards, M.G., Ross, A.M., Hastings, D.E., and Rhodes, D.H., "Empirical Validation of Design Principles for Survivable System Architecture," 2nd Annual IEEE Systems Conference, Montreal, Canada, April 2008.

[12] Rader, A.A., Ross, A.M., and Rhodes, D.H., "A Methodological Comparison of Monte Carlo Methods and Epoch-Era Analysis for System Assessment in Uncertain Environments," 4th Annual IEEE Systems Conference, San Diego, CA, April 2010.

[13] Barbará D, Li Y, Couto J., "COOLCAT: an entropy-based algorithm for categorical clustering," Proceedings of the eleventh international conference on Information and knowledge management. ACM, New York, NY, 2002:582–589.

[14] Gan, Guojun, Chaoqun Ma, and Jianhong Wu, *Data Clustering: Theory, Algorithms, and Applications*, ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia, ASA, Alexandria, VA, 2007.