SYSTEM RISK VISUALIZATION AND MITIGATION METHODOLOGY AND ITS APPLICATION TO ICT SYSTEM FAILURES

Takafumi Nakamura
Fujitsu FSAS Inc., System Support Promotion Unit, Nomura Fudosan Musashikosugi Bldg.-N, 13-2, Nakamaruko, Nakahara-ku, Kawasaki-shi, Kanagawa 211-0012, JAPAN

ABSTRACT

A method is presented for mitigating system failures. Current state-of-the-art methodologies and frameworks have strength as a common language to understand system failures holistically with various stakeholders. On the other hand there is a shortcoming in quantitative aspects. This is major obstacle to assess effectiveness of various measures to mitigate system risk. In order to overcome this shortcoming, this paper express system risk numerically through a coupling and an interaction factors between system configuration elements as well as system failures frequency rate, this three numerical number (i.e. coupling, interaction and frequency) create three dimensional space, and measuring its trajectory through time visualize system risk trends which are the targets to create an effective preventative measures to system failures. A root cause of a system failure is discovered by using a System Dynamics technique to a trajectory of a system risk location, then based upon the root cause, the effective counter measures are extracted. Lastly this methodology is applied to the system failures cases with various ICT systems and counter measures are extracted. An application example of ICT system failures exhibits the effectiveness of this methodology.

Keywords: Risk management; Crisis management; Normal Accident Theory (NAT); High Reliability Organization (HRO); Information and Communication Technology (ICT); System Dynamics

1. INTRODUCTION

There are many examples of similar system failures repeating and of negative side effects created by quick fixes. Introducing safety redundant mechanisms does little to

reduce human errors. As pointed out by Perrow (1999, p. 260), the more redundancy is used to promote safety, the greater the chance of spurious actuation; "redundancy is not always the correct design option to use." While instrumentation is being improved to enable operators to run their operations more efficiently, and certainly with greater ease, the risk would seem to remain about the same.

Weick and Sutcliffe (2001, p. 81) explained why traditional total quality management (TQM) has failed. "We interpret efforts by organizations to embrace the quality movement as the beginning of a broader interest in reliability and mindfulness. But some research shows that quality programs have led to only modest gains…this might be the result of incomplete adoption. But we would go even further, and argue that the reason for incomplete adoption is the necessary infrastructure for reliable practice…is not in place even where TQM success stories are the rule. The conclusion is consistent with W.E. Deming's insistence that quality comes from broad-based organizational vigilance for problems other than those found through standard statistical control methods."

From the other perspectives, there are six stages from the initial stage to cultural readjustment through catastrophic disasters (Turner et al., 1997, pp. 88). They are Stage I: Initial beliefs and norms, Stage II: Incubation period, Stage III: Precipitating event, Stage IV: Onset, Stage V: Rescue and salvage, and Stage VI: Full cultural readjustment. The second stage, or incubation period, is hard to identify because of the various side effects of quick fixes (Turner et al., 1997 and Nakamura et al., 2010). Therefore, the second stage plays the crucial role that leads to catastrophic disaster. Many side effects due to quick fixes of information and communication technology (ICT) systems have been identified (Nakamura et al., 2009a, 2010). There are two factors in particular that make it difficult to prevent ICT system failures: the lack of a common language for understanding system failures and the lack of a methodology for preventing future system failures. These shortcomings result in local optimization and the introduction of quick fixes as countermeasures.

This paper aims to mitigate system failures by promoting holistic as well as quantitative approach and by introducing a system risk visualization methodology. This approach is novel in that current methodologies tend to focus only on static nature to understanding system failures and tend to lose emergent nature arising over time, which leads to myopic management and fails to recognize invaluable ways to recognize the shortcomings of state-of-the-art methodologies (i.e., current methodologies fail to change the status quo).

2. VARIOUS METHODOLOGIES AND THEIR STRENGTH AND LIMITATION

In this chapter reviews current four methodologies or framework. They are SOSF, IC chart, Human error framework and Close code metrics.

2.1 SOSF meta-methodology (System of system failure)

The proposed system of system failure (SOSF) meta-methodology for covering all system failure models (Nakamura et al., 2009a, 2007 and 2010) is derived from the system of system methodologies (SOSM) (Jackson 2003 and 2006) and system failure classes. The system of system methodologies classifies the world of objects into two dimensions: systems and participants. The system dimension has two domains: simple and complex. The participant dimension has three domains: unitary, plural, and coercive. Therefore, SOSM classifies the world of objects into six (2 × 3) domains, and there is an appropriate methodology for each domain. The system of system failure complementarily covers these domains on the basis of this worldview to enable the viewing of object system failures. SOSF uses four domains (excluding the coercive domain because the main focus of this paper is technological systems rather than broader social domains) from SOSM. On top of these four domains, we add a third dimension to identify the person or factor responsible for the system failure. To identify the root causes of failures, we classify system failures on the basis of system boundaries and the responsible system level introduced with the viable system model (VSM) (Beer, 1979 and 1981). Failures are classified in accordance with the following criteria (Nakamura et al., 2009a, 2009b and 2010).

Class 1 (Failure of deviance): The root cause is within the system boundary, and conventional troubleshooting techniques are applicable and effective.

Class 2 (Failure of interface): The root cause is outside the system boundary but is predictable in the design phase.

Class 3 (Failure of foresight): The root cause is outside the system boundary and is unpredictable in the design phase.

System safety can be achieved through the actions of various stakeholders. One such common language was developed by Van Gigch (1986) for the taxonomy of system failures. There are six categories of system failures, i) technology, ii) behavior, iii) structure, iv) regulation, v) rationality, and vi) evolution. In particular, SOSF was designed by allocating each type of failure from this taxonomy (Van Gigch, 1986) into an SOSM meta-methodology space. Fig. 1 shows this space.

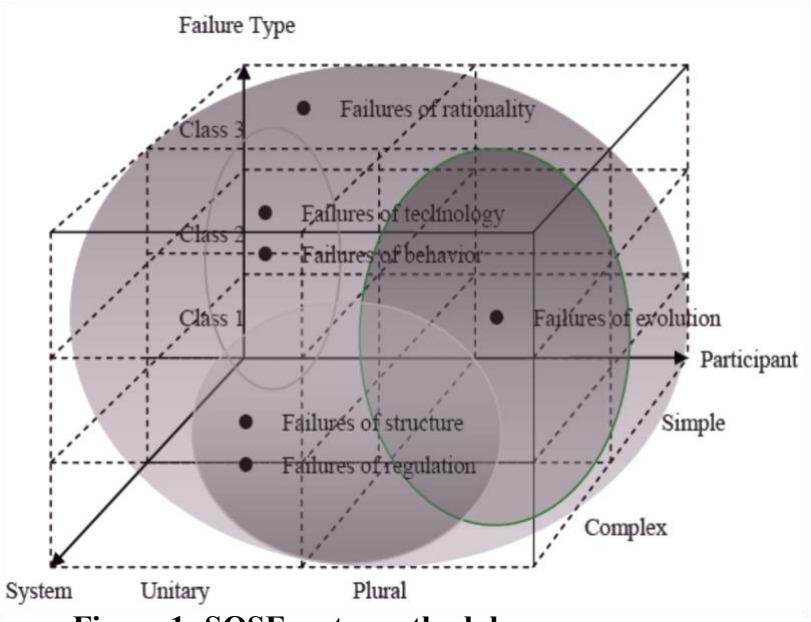

Figure 1: SOSF meta-methodology space

There are two widely used failure analysis techniques: failure mode effect analysis (FMEA: IEC60812, 2006) and fault-tree analysis (FTA: IEC61025, 2006). FMEA deals with single-point failures by taking a bottom-up approach, and is presented as a rule in the form of tables. In contrast, FTA analyzes combinations of failures in a top-down way, and is visually presented as a logic diagram.

Both methodologies are mainly used in the design phase. However, these methodologies are heavily dependent on personal experience and knowledge, and FTA in particular has a tendency to miss some failure modes in failure mode combinations, especially emergent failures.

The major risk analysis techniques (including FMEA and FTA) are explained in (Bell, 1989, pp. 24–27; Wang et al., 2000, Chapter 4, and Beroggi et al., 1994). Most failure analyses and studies are based on either FMEA or FTA. FMEA and FTA are rarely both performed, though, and when both are done they will be separate activities executed one after the other without significant intertwining.

Current methodologies tend to fail to take a holistic view of the root causes of system failures. And a majority of them stay in the unitary-simple-class 1 domain. It is important to identify and cover the plural-complex-class 3 domain. In the next section, another method is introduced for understanding system failures holistically.

2.2 Normal accident theory and IC chart

It is not unusual that several failures happen sequentially or simultaneously. Each is not a catastrophic failure in itself; however, the complex (i.e., unexpected) interaction

of those failures may have catastrophic results. Tight coupling of a component involves a cascade of single-point failures that quickly reach a catastrophic end before safety devices come into effect. This is called system failure or a normal accident as opposed to a single-point failure. Perrow (1999) analyzed system failures using the interaction and coupling of system components. This is called normal accident theory.

The IC chart is a table for classifying object systems by interaction and coupling. Fig. 2 shows the IC chart developed by Perrow (1999). Topological expression was done subjectively by Perrow (1999). By combining the two variables in this way, a number of conclusions can be made. It is clear that the two variables are largely independent. Examine the top of the chart from left to right. Dams and nuclear plants are roughly on the same line, indicating a similar degree of tight coupling. But they differ greatly on the interaction variable. Whereas there are few unexpected interactions possible in dams, there are many in nuclear plants. Or, looking across the bottom, universities and post offices are quite loosely coupled. If something goes wrong in either of these, there is plenty of time for recovery, and things do not have to be in a precise order. But in contrast to universities, post offices do not have many unexpected interactions—it is a fairly well laid out (linear) production sequence without a lot of branching paths or feedback loops. The IC chart defines two key concepts, the types of interaction (complex and linear) and the types of coupling (loose and tight). They are laid out so that we can locate organizations or activities that interest us and show how these two concepts, interaction and coupling, can vary independently of each other.

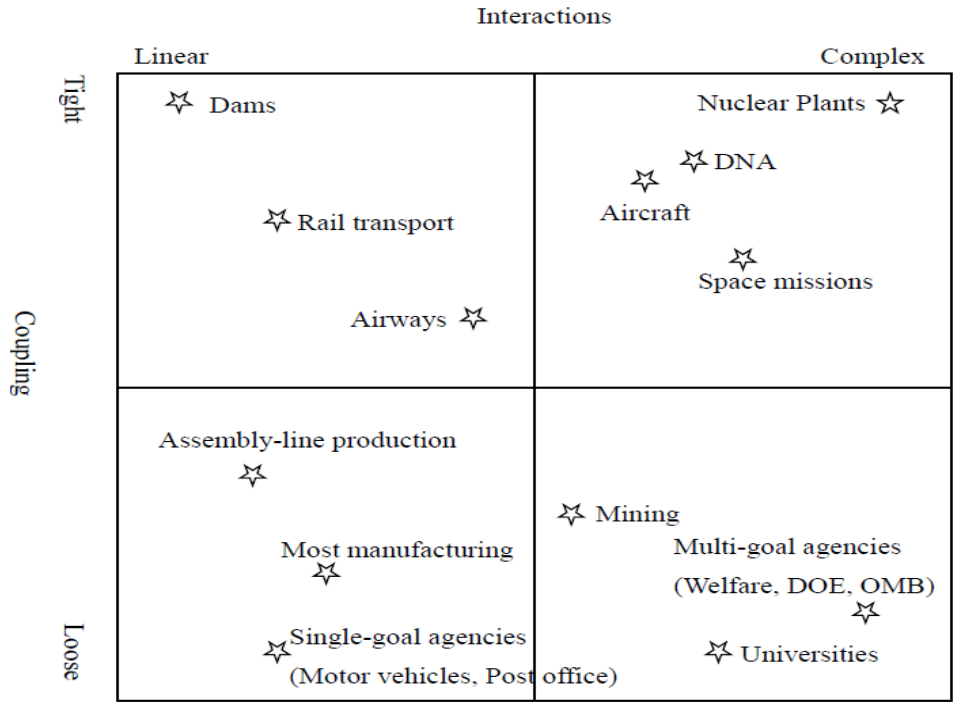

Figure 2: Interaction/Coupling Chart

2.3 System failure framework

Partial solutions are not enough to promote safety for"safety is a system problem (Leveson, 1995, 2004 and 2009)." To solve the safety issue, this paper introduces a system failure framework to accommodate the holistic perspective. It consists of two basic dimensions: the horizontal, which pertains to the scope or size of a problem or situation that a person is inherently (instinctually) comfortable in dealing with, and the vertical, which pertains to the kind of decision-making processes that a person inherently (instinctually) brings to bear on a problem or situation. The framework is important because it shows that, for the how and why on any issue or problem of importance, there are at least four very different attitudes or stances with regards to the issue or problem. None of them is more important or right, so we need to check all perspectives intentionally in order to overcome psychological blind spots. Fig. 3 shows the framework.

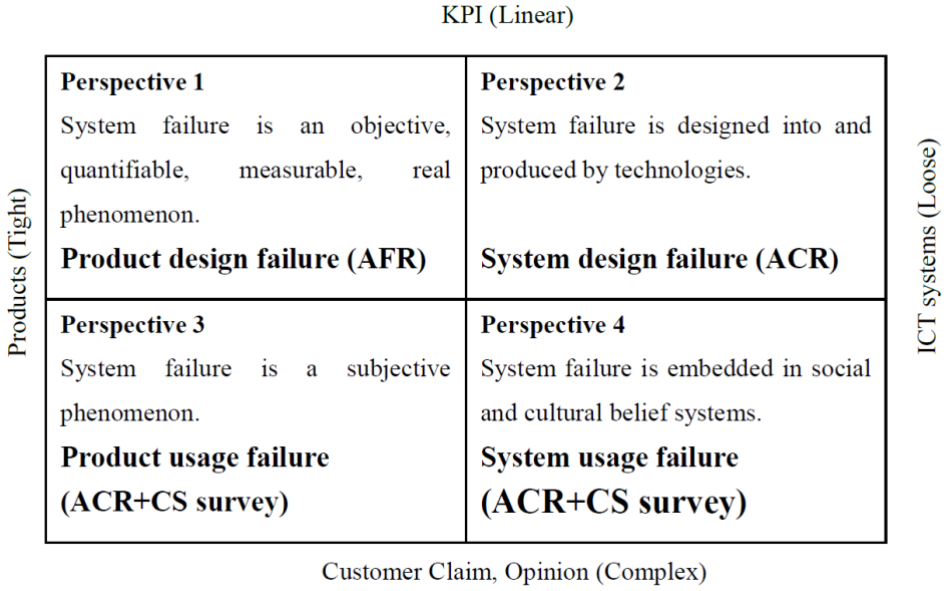

Figure 3: System failure framework

2.4 Close code metrics

Our proposed method for quantifying the risk factors for system failure uses the close code of system failures over time. Close code is the failure root case classification taxonomy (hardware, software error, human error, etc.) used for systems. In many industry and organization, system failures are classified using close codes based on the root cause analysis of failures. Here we use the close code system as a metric for objectively representing risk.

Generally, close codes are classified into two dimensions. The first dimension consists of phases for creating an object system (i.e., design, configure, and operate in time sequence) and the second is the nature of the stakeholders (i.e., simple or complex) responsible to system failures. The close code system is a filter of the root causes of system failures. Figure 4 illustrates the general concept of close code classification. The loop represents learning cycles, and, in complex cases, the learning cycles are spread over multiple phases and stakeholders. Most industries use the close code system reactively for single-system failures. However, it is important to monitor the close codes accumulated for any arbitrary amount of time and to identify effective countermeasures. This requires introducing metrics to quantitatively represent the risk status.

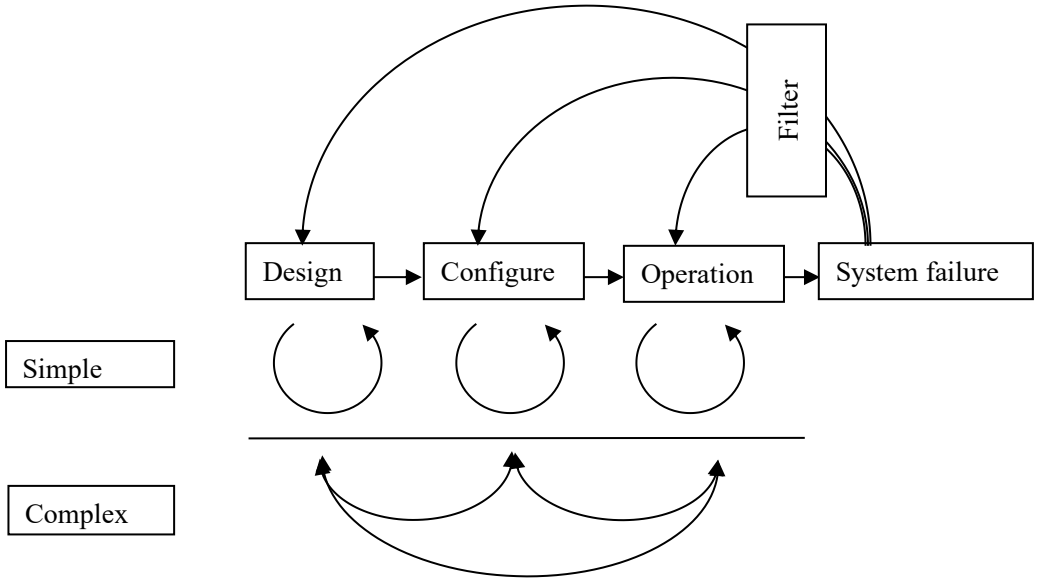

Figure 4 Classification of close codes and learning cycle from system failure

The close code system should be checked in terms of the taxonomy of system failures (Van Gigch; 1986, 1991). This is done to verify that the close code system is mutually exclusive and collectively exhaustive. Table 1 is an example mapping of a close code system onto a close code matrix for the ICT industry. While the close code system varies by system and industry, the close code system is classified into a close code matrix with the two dimensions.

The two-tuple number (X, Y) represents the domain in the close code matrix. For example (3, 2) represents the operation-complex domain. The symbols A, B, P, T, and N represent the causes of system failure: A means hardware malfunction, B means human behavior error, P means maintenance period expiration, T means target products in question is not supported due to other vendors and N means future consideration to implement new features (i.e., to avoid further system failures). Causes A and B have subcategories: A1 means CPU, A2 means memory, A3 means channel, A4 means power, A5 means disk, AB means hardware setup mistake, A6 means other IO, and AU means unknown cause; BA means network setup mistake, BB means IO setup mistake, BC means parameter setup mistake, BD means installation mistake, BE means operation mistake, BF means application coding mistake, and BG means other mistake.

The close code matrix is related to the IC chart in terms of system failure classification. The first dimension of the close code matrix (design, configure, and operate) corresponds to the interaction axis of the IC chart. The second dimension (simple and complex) corresponds to the coupling axis. The next section introduces the metric

derived from the close code matrix for use in the SOSF space.

Table 1 Mapping close code system onto close code matrix

	Close Codes	1 (Design)	2 (Configure)	3 (Operation)
		Failure of Technology and Structure	Failure of regulation	Failure of behavior and evolution
		Failure of Rationality, Evolution		
1 (Simple)	A (Hardware)	A(1~5)		A(B)
	B (Behaviors)		B(A~D,F)	B(E)
	P (Obsolete)		P	
2 (Complex)	A (Hardware)	A(6)		A(U)
	B (Behaviors)			B(G)
	T (Other vendors not Supported)		T	
	N (Future plan)	N		

Table 2: Isomorphic structure between three perspectives

	IC chart	System failure framework	System of system failure	Close code matrix
Vertical Axis	Linear	KPI	Simple	Simple
	Complex	Customer Claim, Opinion	Complex	Complex
Horizontal Axis	Tight	Product	Unitary	Design
	Loose	ICT systems	Plural	Operation

So far, this paper has introduced four perspectives to promote system safety. An isomorphic structure is depicted between the four perspectives to promote further holistic views. Table 2 shows the isomorphic structure. The combination of these four perspectives promotes various perspectives to learn from previous system failures.

3. PROPOSAL FOR NEW METHODOLOGY

3.1 The challenge of the state-of-the-art methodologies

The previous section introduced various methodologies and frameworks, they have strength as a common language to understand system failures holistically with various stakeholders. On the other hand there are several shortcomings of measurement quantitative aspects. They are 1) clarify risk migration direction over time, 2) hinder effective discussion about current risk situation and 3) clear understanding a nature of system failures but weak connection to counter measures to mitigate system failures. The new methodology should overcome above mentioned shortcomings. The way to reach new methodology have two features. One is to focus isomorphic structures between current state-of-the-art methodologies to extract basic structure for visualizing system risk. The other is to introduce quantitative measure for system risk. The three dimensional system risk space enables us to monitor system risk location (i.e. SRL) in system risk space (Nakamura et al., 2014).

3.2 Topological presentation of system failure risk factors

We introduced the system failure space (SOSF) in Section 2 and introduced the quantification of risk factors on the basis of the close code system in Section 2.4. By using the close code matrix as the metric in the SOSF, we can topologically present the risk factors for system failure. Every single-system failure is located in the SOSF space. Object system risk location is presented topologically within the SOSF space with this metric. An object system's risk factor is represented quantitatively in the SOSF space with the metric, which tracks the transition of the risk factor over time. The risk factor location for an object system at any arbitrary time is represented by a three-tuple number. The object system risk location in the SOSF space is represented by system risk location (SRL) (X, Y, Z), where X represents the metrics of system interactions, Y represents the metrics of system coupling, and Z represents the annual call rate (ACR: incidents/100 shipments per year).

There are several steps for introducing these metrics into the SOSF space, as shown in Fig. 5. The first step is defining a system failure group at any arbitrary time. This group is the basis used for calculating system failure risk factors and is expressed in the SOSF space. The second step is mapping the corresponding close code system onto the closed code matrix. The third step is matching the close code matrix to the IC chart.

The X (Y) axis corresponds to the interaction (coupling) axis. The (3, n) (n=1: simple; n=2: complex) area in the close code matrix corresponds to the right side of the

interaction axis (i.e., complex area) in the IC chart. The (m, 2) (m=1: design; m=2: configuration; m=3: operation) area in the close code matrix corresponds to the lower area of the coupling axis (i.e., loose area) in the IC chart. The quantification of risk factors is achieved using the (m, n) notation in the close code matrix. The complex interaction risk factor is represented by α: (3, n)/ number of system failures at any arbitrary time. The loose coupling risk factor is represented by β: (m, 2)/ number of system failures at any arbitrary time.

Fig. 5 shows that β is the area inside α; therefore, β is defined as (3, 2)/number of system failures, not (m, 2)/number of system failures. The reason for measuring β in α is that the risk of an object system should be measured during operation. The complex and loose risk factors of an object system is represented as a two-tuple number: γ = (α, β). This is the quantitative coordinate point in the IC chart.

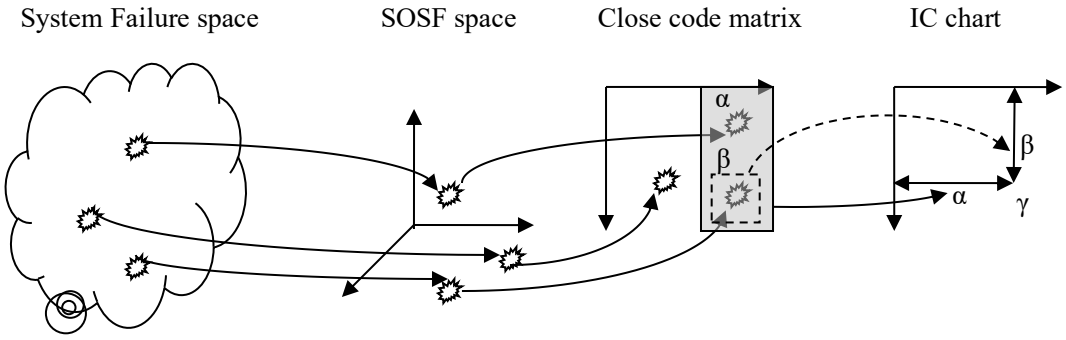

Figure 5 General Sequence of introducing metrics into SOSF

We define γ as the representation of an object system's risk factor and objectively place it in the IC chart by using a metric. Fig. 6 gives a detailed explanation of γ = (α, β) on the IC chart from the system failure group at any arbitrary time. Adding a new dimension (i.e., the Z axis representing ACR) to γ produces an SRL (α, β, ACR). The γ can only represent the looseness and complexity of the target system. The reason for introducing a 3rd dimension, ACR, is that the frequency of system failure should be incorporated in the system-failure metric.

ICT development engineers use the annual failure rate (AFR) for monitoring system component quality rather than system quality. ICT users who encounter a problem with a product report it to the help desk, and the help desk provides them with a solution. The help desk then identifies the cause of the problem, and, if it was due to faulty product design, the help desk escalates it to the development section for further investigation. The development section designs new products on the basis of the data

for the escalated problems. User-reported problems are screened at the help desk so that the development section can concentrate on product-related issues. The development section measures product quality on the basis of the AFR using only the problems escalated from the help desk, not on the basis of the ACR using all problems reported by the users. The metric for product quality is the AFR, and that for system quality, which includes product quality, is the ACR. They are calculated as shown in Fig. 7.

Matching the 3rd axis of the SOSF area (i.e., system failure classes 1, 2, and 3) with the ACR is straightforward because the AFR corresponds to class 1 failures. The difference between the ACF and AFR corresponds to class 2 and 3 failures. Fig.6 shows the transition of SRL over time. To emphasize the magnitude of the ACR, the size of the black circles in this figure changes according to the ACR value. Fig. 6 also shows that the initial SRL at t0 could shift to the SRL at t1 with increasing ACR (large circle) or to the SRL at t2 with decreasing ACR (small circle). The black Circles in the metric SOSF space change size to represent the ACR. Figure 10 shows the transition to linear and tight with deceasing ACR. Table 7 summarizes the metrics introduced into the SOSF space and the relationship between the closed code matrix and IC chart.

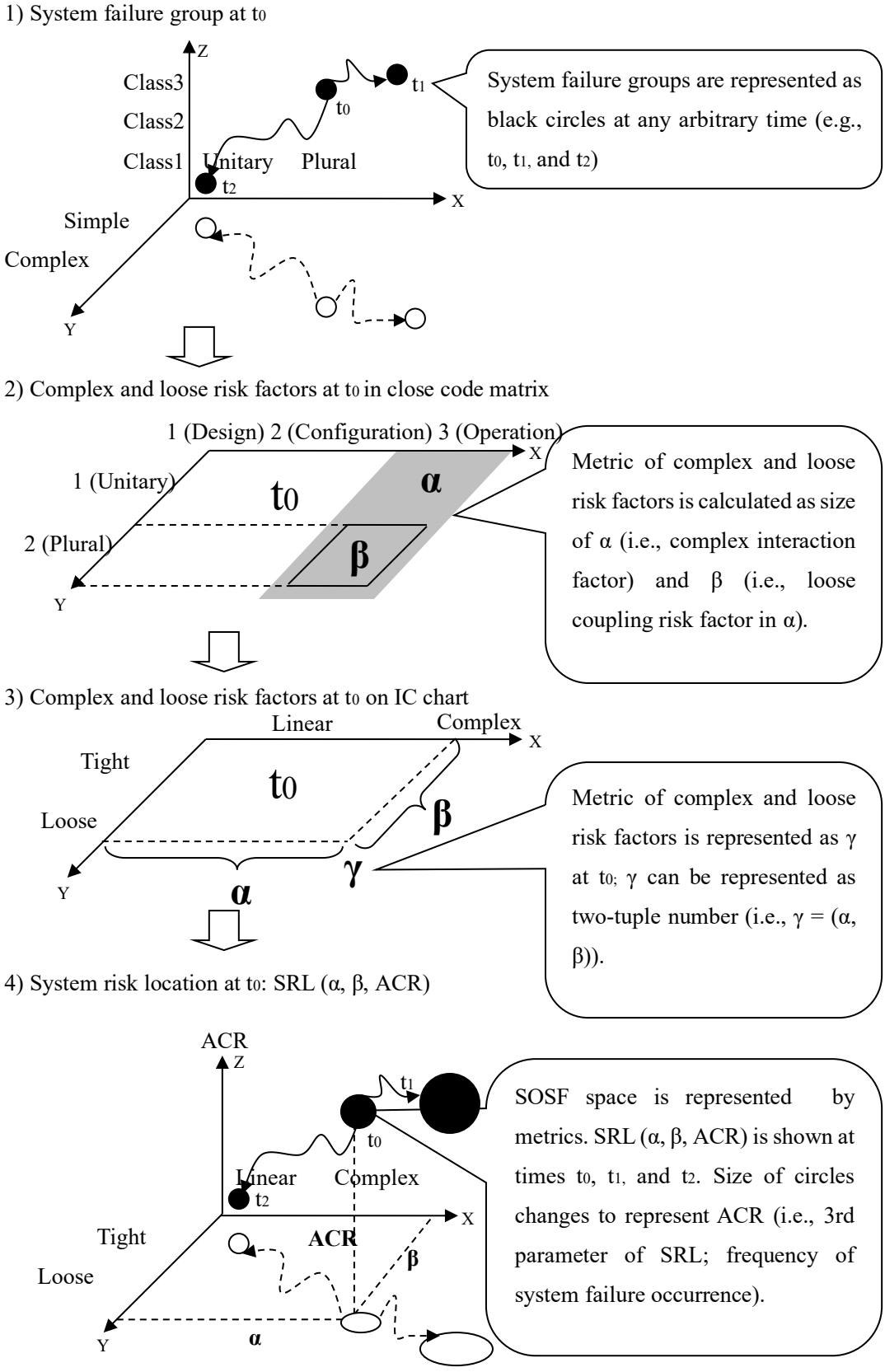

Figure 6 Detailed diagram of metric generation

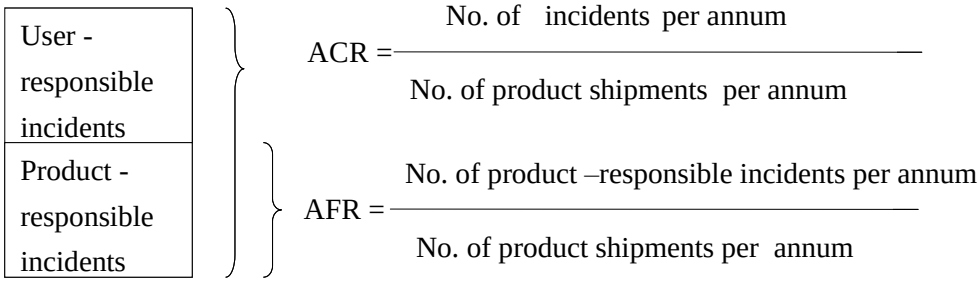

Figure 7 Calculation of annual failure rate (AFR) and annual call rate (ACR)

Table 3 Summary of SOSF to SOSF with metrics via close code matrix and IC chart

Axis	SOSF	Close code matrix	IC chart	Metric SOSF SRL (X,Y,Z)
X	Stakeholders (i.e. Unitary and Plural)	System Creation Phase (i.e. Design, Configuration and Operation)	Interaction (i.e. Linear and Complex)	Interaction Metrics(α): (3,n)/all incidents
Y	System feature (i.e. Simple and Complex)	System feature (i.e. Simple and Complex)	Coupling (i.e. Tight and Loose)	Coupling Metrics (β): (3,2)/all incidents
Z	Failure class (i.e. Class1,2 and 3)	N/A	N/A	ACR (including AFR)

According to the above discussion of an SRL (α, β, ACR), a larger α:(3,n) indicates that the object system has more complex interaction, a larger β:(n, 2) indicates that the object system has looser coupling, and a larger (3,2) indicates that the object system has more complex and looser properties.

3.3 Structure of System Risk visualization methodology

Step1 and 2 are the phase to understand system risk situation and its trend over time. Step3 scrutinizes system risk dynamic structures focusing on complex, linear, Loose and tight factors derived at Step2. Step4 is the phase to surfacing reinforcing unintended consequences loop (to increase system failures) as well as balancing intended consequences loop (to mitigate system failure).Last step is to find out the way to mitigate system failures at Step5. Fig. 8 is the sequence of this new methodology.

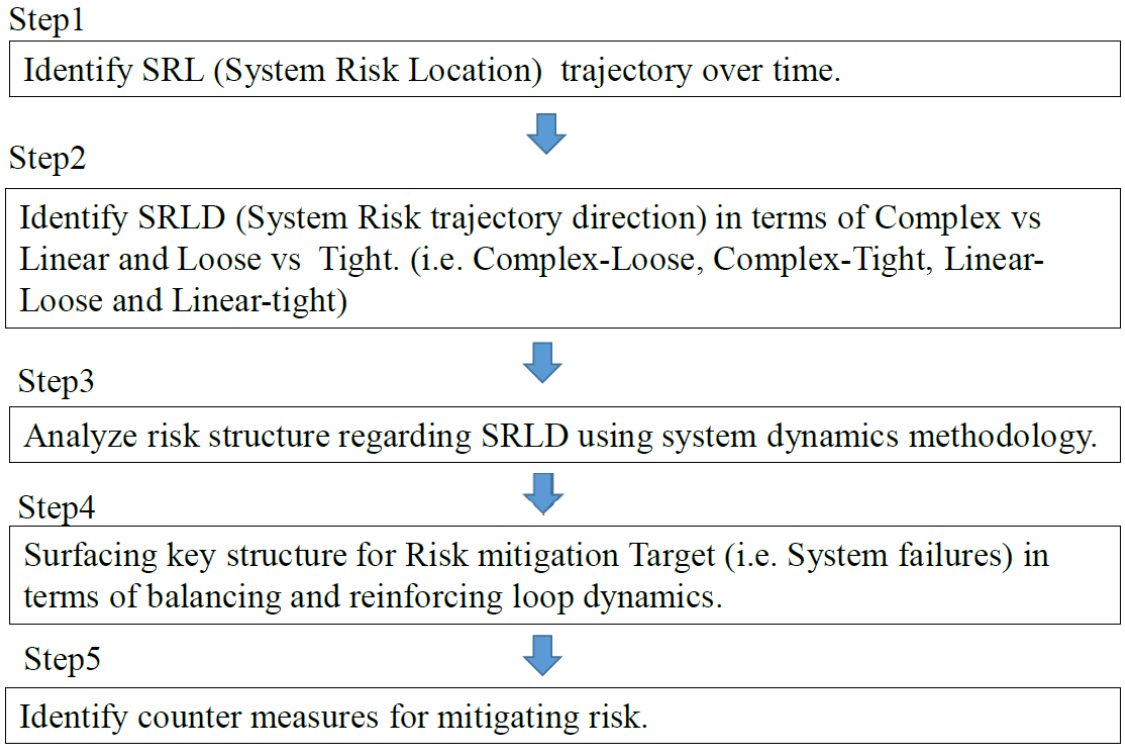

Figure 8 Five steps sequence of new methodology

4. APPLICATION ICT SYSTEM FAILURES

4.1 Current ICT methodologies

Computing systems are characterized by five fundamental properties: functionality, usability, performance, cost, and dependability (Avizienis et al., 2001). The dependability of a computing system is the ability to deliver service that can justifiably be trusted (Laprie, 1992). This property integrates six basic attributes: reliability, availability, safety, confidentiality, integrity, and maintainability. Conventional development models, either for hardware or for software, do not explicitly incorporate all the activities needed for the production of dependable systems. Indeed, while hardware development models (e.g., BSI, 1985) traditionally incorporate reliability evaluation, verification, and fault tolerance, traditional software development models, e.g., Waterfall (Royce, 1970), Spiral (Boehm, 1986), and V (Forsberg et al., 1991), incorporate only verification and validation activities and do not include reliability evaluation or fault tolerance. Several models have been proposed (Kaaniche et al., 2002). Those models are explicitly incorporated in a development model focused on the production of dependable systems.

4.2 Research methodology

The methodology used in this research was quantitative. The factors contributing to ICT systems failures were analyzed and the symptoms were examined. Since the author is the "owner" of the research projects, we can carefully eliminate other elements to influence the outcome of this research as much as possible and to confirm that the conclusion of the research was not the results of other changes in the studied systems and their management structures.

1) Data gathering and analysis

Every system failure was identified from incidents reported by the field operation group in an ICT company where the author belong. Data on those relating to the researched ICT systems were collected and analyzed quantitatively.

2) Close code metrics formulation

The close code metrics were identified on the basis of the author's experience in the ICT engineering arena. The same metrics were used throughout the research period.

4.3 Application ICT systems failures

The application target system is Virtualized ICT systems which are composed form Operation system (i.e. OS), Virtualization plat form and Network. Fig. 9 shows the overview of Virtualized ICT systems. The system risk factor distribution for Virtualized ICT systems is shown in Table 4, The SRL (α, β) was calculated on the basis of the incidents that occurred in the corresponding system components (i.e. OS, Virtual platform and Network).

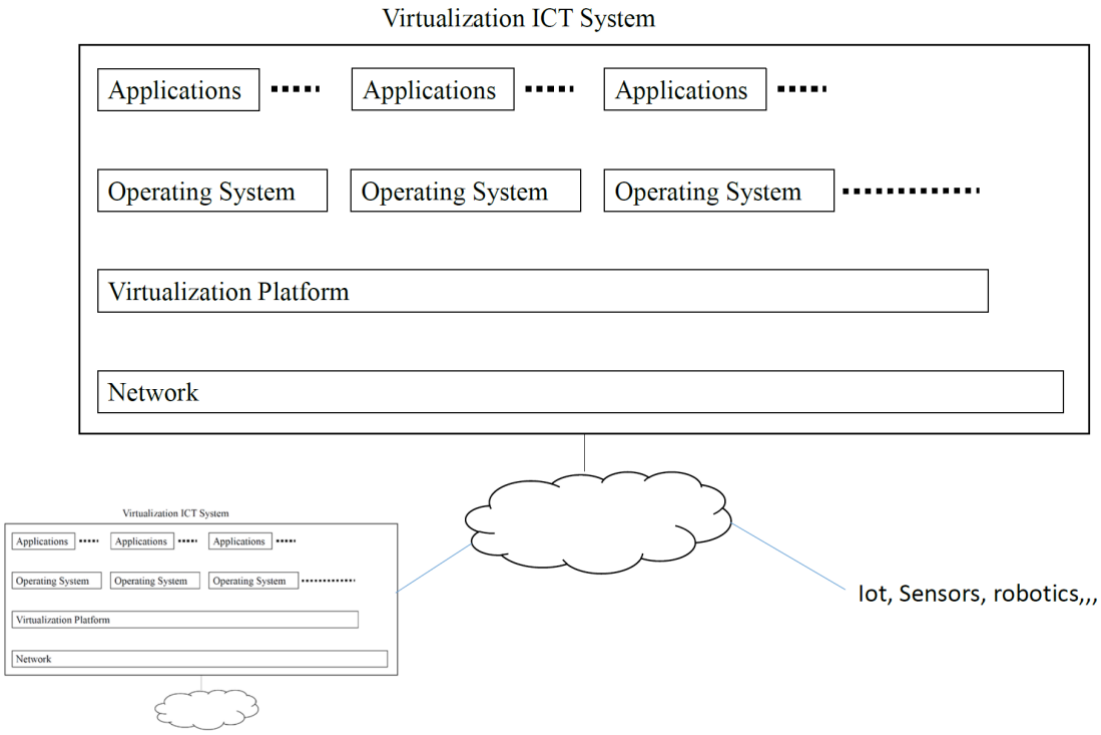

Figure 9 Virtualization ICT systems overview

The new methodology consist five steps and following are the explanation on application step by steps.

Step1 Identify SRL (System Risk Location) trajectory over time utilizing SRL.

Table 4 is the transition data of SRL. According to the table 4, Network and Virtualization platform are moving towards complex - loose direction with increasing ACR. OS is moving towards linear – tight direction with decreasing ACR.

Table 4 Transition of SRL (System Risk Location)

		α			β			ACR		
		2016	2017	2018	2016	2017	2018	2016	2017	2018
OS	Total	19.3	19.5	18.1	19.1	19.2	18.0	15.5	20.6	17.6
	Operation	13.1	13.4	12.7	12.0	12.3	11.4	3.9	5.3	4.2
Virtual	Total	16.7	18.0	20.3	16.4	16.8	19.2	31.2	39.6	41.7
	Operation	13.9	14.6	17.6	12.7	13.3	16.2	8.3	11.0	14.1
Network	Total	12.6	13.0	15.9	8.9	8.7	12.1	44.6	57.2	52.0
	Operation	2.6	3.1	3.3	1.9	2.2	2.3	1.9	3.0	2.9

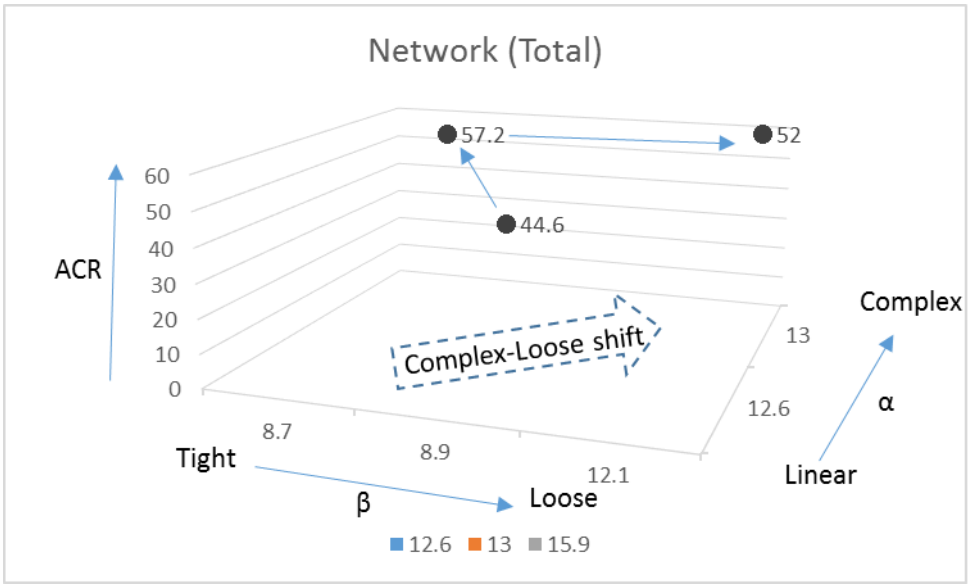

Figure 10 Network SRL trajectory

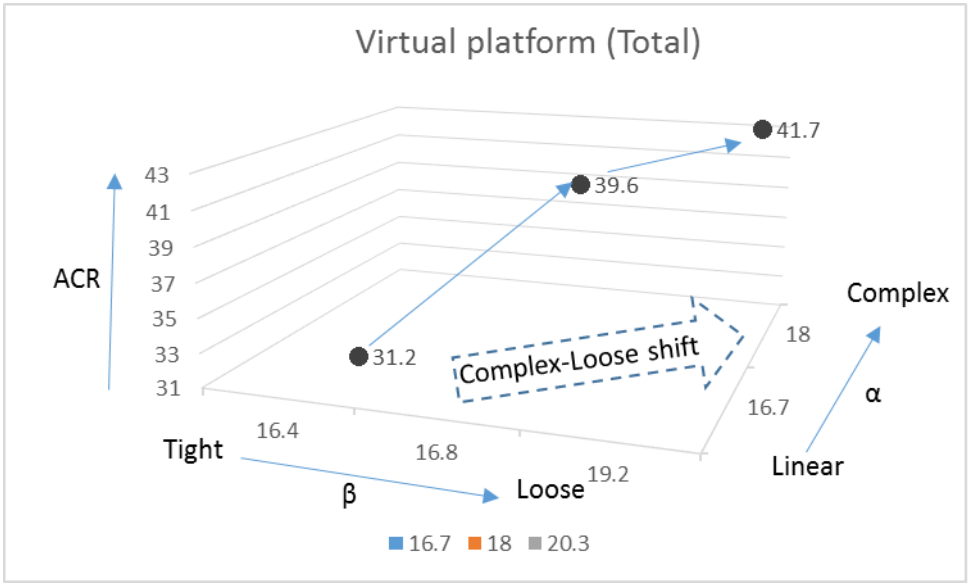

Figure 11 Virtual platform SRL trajectory

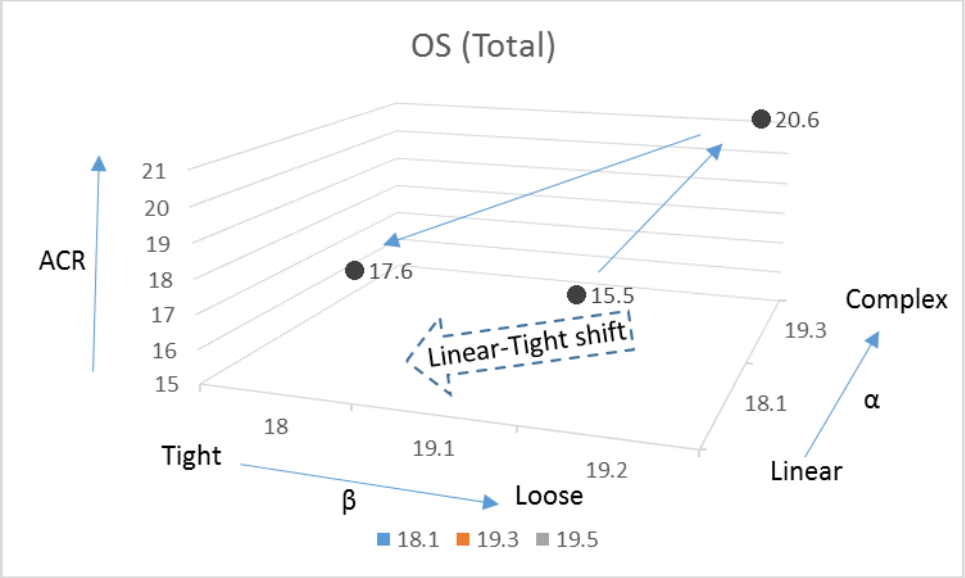

Figure 12 OS SRL trajectory

Step2 Identify SRLD (System Risk trajectory direction) in terms of Complex vs Linear and Loose vs Tight. (i.e. Complex-Loose, Complex-Tight, Linear-Loose and Linear-tight)

SRLD of Network and Virtualization platform is migrating towards Complex-Loose direction and OS is migrating towards Linear – Tight direction. SRL trajectory of each products is visualized by Fig 10, 11 and 12.

Step3 Analyze risk structure regarding SRLD using system dynamics methodology.

Fig. 14 is the dynamic structure for system failures using system dynamics. Complex and Loose factors are extracted at step 2, and they are used for analyzing dynamic structure. Complex and Loose are the main contributors for system failures. The increasing system failures dynamism relating complex factor are followings three sequences.

- Cost pressure -> Virtualization -> Complex (Interaction) -> System Failure +
- Cost pressure -> Virtualization -> Complex (System) -> System Failure +
- Cost pressure -> Virtualization -> Plural -> Complex (System) -> System Failure +

On the other hand, the decreasing system failures dynamism relating system failure is following one sequence.

- Quality culture ->Standardization ->Loose -> System Failure -

Fig13 is the total picture of system dynamics relating to system failures.

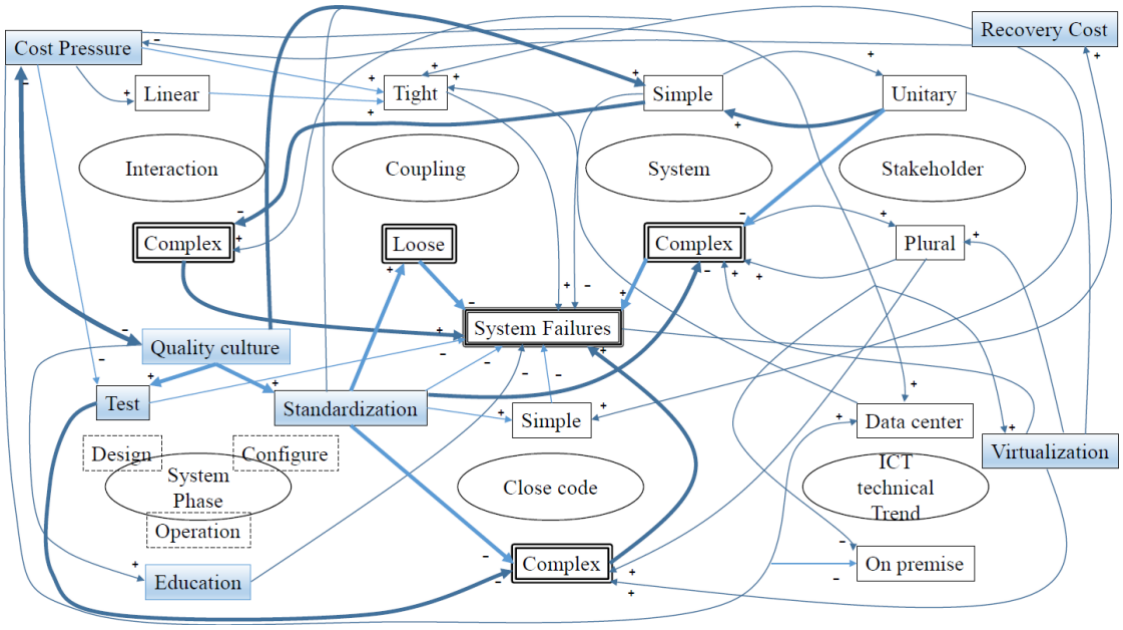

Figure 13 System dynamics relating system failures

We can find several key notations useful for examining systems failures incorporated in conventional dynamic models. Table 5 summarizes the symbols used in dynamic models. R or B is combined with IC or UC, for example BIC stand for Balancing intended consequences loop. The "+" sign indicates an increase (decrease) of a state1 causes an increase (decrease) of state2. The "-"sign indicates an increase (decrease) of a stete1 causes a decrease (increase) of state2. These symbols are used in Fig. 13.

Table 5 the symbols used in a dynamic model

Symbol/Notation	**feature**
R	Reinforcing loop
B	Balancing loop
IC	Intended Consequences with combination of R or B
UC	Unintended Consequences with combination of R or B
+	Positive feedback loop
-	Negative feedback loop

Step4 Surfacing key structure for Risk mitigation Target (i.e. System failures) in terms

of balancing and reinforcing loop dynamics.

Fig. 14 is the diagram regarding RUC and BIC to system failures. Evolution of Virtualization technologies and increase maintaining ICT system cost pressure are creating RUC for System failures.

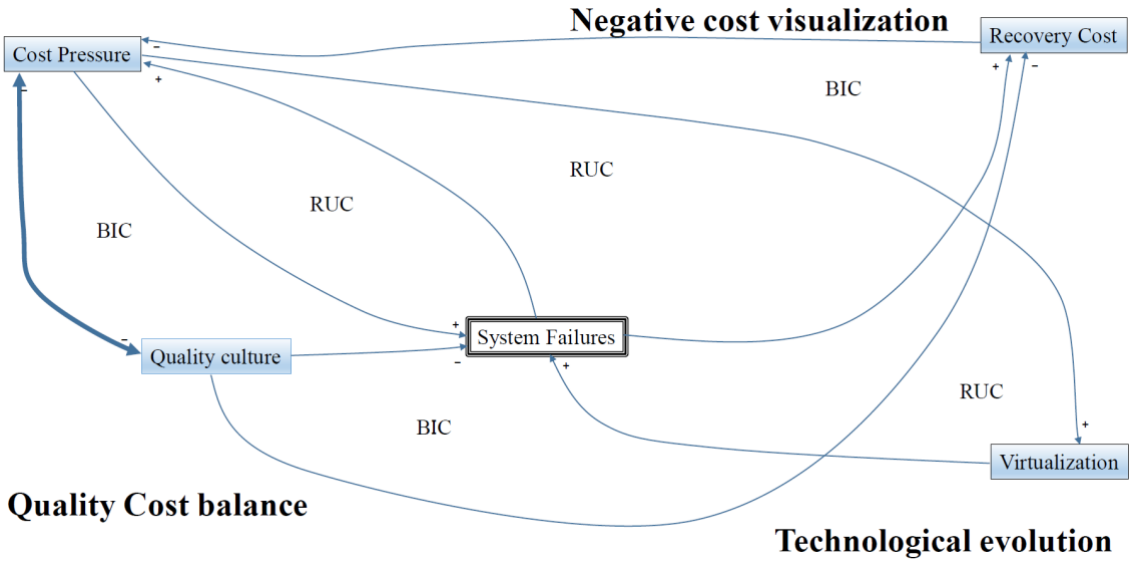

Figure 14 Balancing and reinforcing mechanism for system failures

Step5 Identify counter measures for mitigating risk.
Two counter measures are extracted from the Step 4. (i.e. structures of RUC and BIC)

They are 1) Promoting quality first culture and 2) Visualizing negative cost (i.e. system recovery, business impacts etc.) once system failures happens. Purse cost and quality balance is the key finding of this methodology.

5. CONCLUSION

Gartner identified ten key IT trends for 2012 (Gartner, 2011). Especially there are two major concerns related to ICT technological risk: virtualization and fabric technology. The evolution of virtualization due to cloud computing technology is increasing "compute per square foot." Virtualization is being used to increase computer density and to vertically scale data centers. If used wisely, average server performance can be increased from today's paltry 7% to 12% average to 40% to 50%, yielding huge benefits in floor space and energy savings. This trend is pushing ICT systems towards tighter coupling domains. Gartner defines fabric technologies as the vertical integration of

servers, storage devices, and network systems and components with element-level management software that lays the foundation for optimizing shared data center resources efficiently and dynamically. This trend is pushing ICT systems towards more complex interaction domains. Therefore the balance between the vertical concentration of software and the physical diversification of hardware is crucial for ICT systems risk avoidance.

In this paper, the proposed methodology is applied to the Virtual ICT systems which are composed by mainly three technologies (i.e. Operating system, Network and virtualization platform product).

It is found out that the risk trend of Network and Virtualization platform products are shifting towards Complex-Loose domain, on the other hand Operating system is shifting towards Liner-Tight domain. Above mentioned shift is the result in accordance with the two environmental trends change. One trend is involved stakeholders are increasing very rapidly as the digital technological evolution around Network and virtualization technology arena. This movement is contributing Complex shift. The other trend is an effort to improve robustness of the system. (i.e. durability of the network and server) This movement is contributing Loose shift. On the other hand operating system is continuously improving efficiency and its operating speed by various vendors. This movement is contributing Linear- Tight shift. This findings is underpin Gartner's technological trends.

In step3 the risk structure is analyzed through Complex and Loose factors. This analysis surfacing two key structures for mitigating system failures. One is the evolution of virtualization technology, which is the main source of creating unintended reinforcing consequences. (i.e. increase system failures) The two counter measures are extracted by finding Balancing intended consequences. (i.e. mitigate system failures) One counter measure is to enhance quality first culture and the other one is to visualize recovery cost if once system failure happens. Those counter measures are relating not only ICT engineers but also CEOs of various companies and Social awareness of efficiency and quality of complex systems.

Through application to ICT system, the proposed methodology shows that it extracts counter measures to mitigate system failures.

Lastly, according to Gartner identified ten key IoT (i.e. Internet of Things) trends for 2019 (Gartner, 2019), Social issues and user experiences are the most intriguing among them. This inevitably promote the complex and loose shift as well as from ICT components products to Service systems shift with wider and deeper. This trend requires more refined method to cope with new system failures and further research are required that it actually mitigate system failures over times.

REFERENCES

Avizienis, A., Laprie, J.C., Randell, B. (2001). Fundamental Concepts of Dependability (LAAS-CNRS Report No. 01145).

Beer, S. (1979). The Heart of Enterprise. John Wiley & Sons: London and New York.

Beer, S. (1981). Brain of the Firm, 2nd edition. John Wiley & Sons: London and New York.

Bell, T.E., ed. (1989). 'Special Report: Managing Murphy's law: engineering a minimum-risk system,' IEEE Spectrum, June, pp. 24-57.

Beroggi, G.E.G., Wallace, W.A. (1994). 'Operational Risk Management: A New Paradigm for Decision Making,' IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 10, October, pp. 1450-1457.

Boehm, B.W. (1986). A Spiral Model of Software Development and Enhancement, ACM SIGSOFT Software Engineering Notes, ACM, 11(4): pp. 14-24.

BSI, 1985. Reliability of Constructed or Manufactured Products, Systems, Equipment and Components, Part 1. Guide to Reliability and Maintainability Programme Management (Report No. BS 5760). British Standard Institution.

Forsberg, K., Mooz, H. (1991). The Relationship of System Engineering to the Project Cycle, Proceedings of the First Annual Symposium of National Council on System Engineering, pp. 57–65.

Gartner: 10 key IT trends for 2012; http://www.networkworld.com/community/blog/gartner-10-key-it-trends-2012, accessed 7 April 2019.

Gartner: Top 10 IoT trends for 2019 and beyond; https://www.networkworld.com/article/3322517/a-critical-look-at-gartners-top-10-iot-trends.html, accessed 7 April 2019.

IEC 60812 (2006). Procedure for failure mode and effect analysis (FMEA).

IEC 61025 (2006). Fault tree analysis (FTA).

Jackson, M.C. (2003). Systems Thinking: Creative Holism for Managers. John Wiley & Sons: London and New York.

Jackson, M.C. (2006). Creative Holism: A Critical Systems Approach to Complex Problem Situations. Systems Research and Behavioral Science Vol. 23, Issue 5, September/October 2006:647-657.

Kaaniche, M., Laprie, J.C. and Blanquart, J.P. (2002) A framework for dependability engineering of critical computing systems, Safety Science, Elsevier, Issue 9, Vol. 40, pp. 731-752.

Laprie, J.C. (1992). Dependability: basic concepts and terminology, dependable computing and fault-tolerant systems. Springer Verlag, Wien-New York.

Leveson, N. (1995). Software: System Safety and Computers. Addison-Wesley.

Leveson, N. (2004). A new accident model for engineering safer systems, Safety Science 42, pp. 237-270

Leveson, N ., Dulac, N., Marais, K., and Carroll, J. (2009). Moving Beyond Normal Accidents and High Reliability Organizations: A Systems Approach to Safety in Complex Systems.

http://sunnyday.mit.edu/papers/HRO-final.doc, accessed 26 April 2019.

Nakamura, T., Kijima, K. (2007). Meta system methodology to prevent system failures. Proceedings of the 51st Annual Meeting of the ISSS in Tokyo (Aug. 2007).

Nakamura, T., Kijima, K. (2009a). System of system failures: Meta methodology for IT engineering safety. Systems Research and Behavioral Science Vol. 26, Issue 1, January/February 2009: 29–47.

Nakamura, T., Kijima, K. (2009b). A methodology to prolong system lifespan and its application to IT systems. Proceeding of the 53rd Annual Meeting of the ISSS in Brisbane (Jul. 2009).

Nakamura, T. and Kijima, K. (2010). Total system intervention for system failures and its application to ICT systems.

http://journals.isss.org/index.php/proceedings54th/article/viewFile/1436/519, accessed 26 April 2019.

https://www.igi-global.com/article/total-system-intervention-system-failure/58369, accessed 26 April 2019.

http://www.irma-international.org/chapter/total-system-intervention-system-failure/76230/, accessed 26 April 2019.

Nakamura, T. and Kijima, K. (2014). Method for quantifying risk factors for system failure and its application to ICT. Risk Management Vol.14, Issue 4 PP231-271

(Nov. 2014).

Perrow, C. (1999). Normal Accidents: Living with High-Risk Technologies. Princeton Paperbacks: New York.

Royce, W.W. (1970). Managing the Development of Large Software Systems, Proceedings, IEEE WESCON, August 1970, pages 1-9.

Turner, B.A., Pidgeon, N.F. (1997). Man-Made Disasters 2nd edition. Butterworth-Heinemann: UK.

Van Gigch, J. P. (1986). Modeling, Metamodeling, and Taxonomy of System Failures. IEEE Trans. on Reliability, vol. R-35, no. 2, 1986 June: 131-136.

Van Gigch, J. P. (1991). System Design Modeling and Metamodeling. Plenum: New York.

Wang, J.X., Roush, M.L. (2000). WHAT EVERY ENGINEER SHOULD KNOW ABOUT RISK ENGINEERING AND MANAGEMENT. Marcel Dekker, Inc.

Weick K.E. and Sutcliffe K.M. (2001). Managing the Unexpected: Assuring High Performance in an Age of Complexity (J-B US non-Franchise Leadership).