Project Performance International

http://www.ppi-int.com/newsletter/SyEN-035.php

SYSTEMS ENGINEERING NEWSLETTER

brought to you by

Project Performance International (PPI)

SyEN 46 – July 20, 2012

Dear Colleague,

SyEN is an independent free newsletter containing informative reading for the technical project professional, with scores of news and other items summarizing developments in the field, including related industry, month by month. This newsletter and a newsletter archive are also available at www.ppi-int.com.

Systems engineering can be thought of as the problem-independent, and solution/technology-independent, principles and methods related to the successful engineering of systems, to meet stakeholder requirements and maximize value delivered to stakeholders in accordance with their values.

If you are presently receiving this newsletter from an associate, you may receive the newsletter directly in future by signing up for this free service of PPI, using the form at www.ppi-int.com. If you do not wish to receive future SE eNewsletters, please reply to the notifying e-mail with "Remove" in the subject line, from the same email address. Your removal will be confirmed, by email.

We hope that you find this newsletter to be informative and useful. Please tell us what you think. Email to: contact@ppi-int.com.

WHAT'S INSIDE:

PPI Events

Read More…

Quotations to Open On

No man needs sympathy because he has to work. Far and away the best prize that life offers is the chance to work hard at work worth doing.

- Theodore Roosevelt

The things that are measured and tracked and are paid attention to by management are the ones that improve.

- Ralph Young

Feature Articles

Economic Aspects of System-of-Systems Engineering

Massimo Pica

Rome, Italy
massimo.pica (at) libero.it

Fundamentals of System-of-Systems Engineering

According to the International Council on Systems Engineering (INCOSE), the definition of System-of-Systems (hereinafter referred to as SoS) applies to a system of interest whose system elements are themselves systems; typically these entail large scale inter-disciplinary problems with multiple, heterogeneous, distributed systems. These interoperating collections of component systems usually produce results unachievable by the individual systems alone.

Every system in a SoS structure is able to operate in a stand-alone mode, and also to contribute to the achievement of higher-level mission requests. The life cycles of individual systems may show some differences, since integrations or replacements of system elements can take place in order to meet system requirements. If these requirements are taken into consideration, then the effects of integration between different SoS elements have to be properly evaluated in each specific case.

In specialized literature, different definitions for SoS can be found, on the basis of the scope of possible SoS applications, basically reflecting three elements: products (i.e., the characteristics of SoS architecture); processes (design/integration/test); and personnel (lead system integrator).

Some specific characteristics of a SoS can be identified in more detail as follows:

- Operational independence of the systems - Each of the individual systems within a SoS has a “life of its own” and can function acceptably and provide useful service without necessarily interacting with other systems.
- Managerial independence of the systems - The individual systems within a SoS are under different authorities.
- Evolutionary development - The different systems within the SoS are developed and upgraded on uncoordinated schedules.
- Emergent behavior - Some of the behavior of the SoS as a whole is not embodied in any one of the systems within it. Emergent behavior is a direct consequence of having the systems interact; the difficulty is ensuring that the emergent behavior is desirable.

- Geographic distribution - Simply put, the systems within the SoS are not all co-located. While it is highly likely that any significant fielded SoS will have this characteristic, it is by no means obvious that it is a necessary characteristic.

SoS engineering is different from the usual systems engineering. For example, if we consider a SoS as composed of interdependent systems connected to provide a given performance, losing part of the system will cause a significant performance degradation of the entire SoS.

From an economic point of view, the selection of applicable cost models has been found to be influenced by a number of discriminating criteria, a few of which are deemed essential: the SoS stakeholders, the SoS architecture and its lead system integrator(s), and the degree of system component independence in terms of activities required in the life cycle stages for each component. This means that, in order for a cost estimation model to be effectively useful in SoS estimates, it is essential to look first at organizations requiring this sort of information for their strategic goals, especially the system acquirers and the system's user communities. Secondly, having established a lead system integrator, she/he will be responsible not only for selecting SoS architectures but, thereafter, for supervising integration and test activities representing significant cost elements to be properly evaluated. Furthermore, the degree of independence of Life Cycle Management activities at component level will certainly influence the selection of cost models applicable to system software and to the remaining elements of SoS Cost Breakdown Structure, taking into account possible cost overlaps and double countings.

The realization of every SoS involves trade-offs between different solutions and between individual systems' performance. An example of a system of systems would be an aircraft. While the aircraft may be developed as a single system, it could incorporate subsystems developed for other aircraft (for example, the radar from an existing aircraft may be incorporated into the aircraft being developed rather than developing a new radar), so that the new aircraft can be considered as a SoS composed of the airframe, engines, radar, avionics and all other elements necessary to meet the aircraft capability requirements.

From a general Systems Engineering (SE) point of view, the implications of a SoS, compared to "elementary" systems, may typically refer to the following specific attributes: broader technical scope; greater complexity of integration efforts; dynamic and challenging design (especially as regards risk/uncertainty issues and to the emphasis on design optimization); re-configurability of system architectures; peculiarity of SoS simulation and modeling; and rigorous interface design and management.

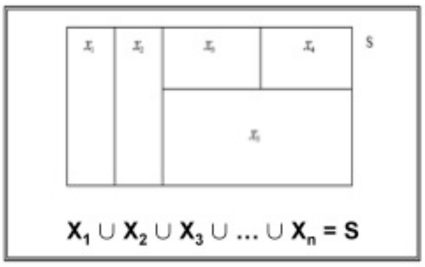
Reverting now to the factors of a more specific economical nature, it is important to emphasize the complex efforts that cost analysts have to undertake to estimate the cost of a SoS, with respect to simpler systems, taking into consideration the full range of options for the overall SoS configuration based on component system selection. Traditional estimation methodologies have to be combined with more advanced methodologies, either analytical or Decision Support methods. Some of these more advanced methodologies will be reviewed.

Bayes Techniques

Henceforth only essential notes will be given on Bayes analytical techniques. They consider an "a priori" hypothesis which is modified by the application of successive information. For a parameter subject to uncertainty constraints, an "a priori" probability distribution will be formulated. For a second value of the parameter, another probability distribution will exist, giving a more refined estimate. Bayesian inference combines these two series of data to obtain an "a posteriori" probability distribution, from which a more likely (or less uncertain value) of the parameter considered can be derived, however such that the actual value will never be the same as the estimated value.

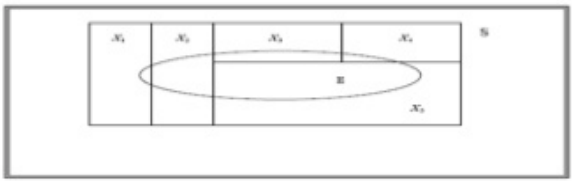
Bayes theorem is used to obtain a subjective probability distribution representing the probability that an event will occur in future trials, taking into account a random variability.

The next figure depicts a series of mutually exclusive events (values) X1, X2, X3,…, Xn, such that X1 X2 X3 … Xn = S



The figure below shows X1, X2, X3… Xn as partitions of the space S, whereas E (being such that E S) is an arbitrary event, which has the following property:

P (E) = P(X1) P (E| X1) + P(X2) P (E| X2) + … + P (Xn) P (E| Xn)



Therefore, if X1, X2, X3… Xn are known, it is possible to determine the probability of event E. While the probability of E cannot be directly observed, it is possible to obtain the conditioned probabilities by applying the Bayes theorem, which also provides inverse probabilities P (Xi|E) from the ratio of P (Xi) P (E| Xi) to P (E).

A vast literature provides more insight into the subject.

Decision Support Methods

The Arrow theorem or "impossibility theorem" is applied whenever it is required to classify in order of importance a series of decision factors, such as lists of technical requirements or of strategic needs. An example will help explain this theorem. Let's suppose that a company needs to adopt an order of preferences between different options. In the company, each individual has some order of preference, which for example can be expressed by a vote. The problem is to find a procedure (for example a suitable voting system), generally referred to as public selection function, which will be able to convert the set of individual preferences into a global consistent order.

Arrow states that if the group of voting citizens includes at least two individuals and the set of available alternatives includes at least three options, it is impossible to create a choice function that will meet at the same time all reasonable requirements for a fair voting system.

In our case, therefore, we can conclude that no analytical technique exists to simultaneously meet all common impartiality criteria in ranking a series of alternatives. Nevertheless, a great variety of circumstances requires selecting and ranking preferences. As a consequence, statistically robust ranking methods should be used, taking into account the inherent limitations of Arrow theorem.

One of such well-known methods is the Analytical Hierarchy Process (AHP), introduced by Professor Thomas Lorie Saaty at the Pittsburgh University. This method can be used to determine the benefit/cost ratio of a system project whenever it is not possible to use solely monetary terms in assessing the benefits and disadvantages deriving from the project implementation, or when it is impossible or difficult for the majority of costs and benefits to operate simulations to quantify them.

The applicability of this method can be better illustrated by the following example: A firm wishes to buy one new piece of equipment of a certain type and has four aspects in mind which will govern its purchasing choice: expense, operability, reliability, and adaptability for other uses (or flexibility). Let's suppose that three options, X, Y and Z, have been offered:

Each of X, Y, and Z will satisfy the firm's requirements to differing extents so which, overall, best meets this firm's needs?

AHP methods are appropriate to address this important and common class of problems and other numerous applications, but also with some limitations.

Saaty established a hierarchical ranking to make a pairwise comparison for the relative assessment of the importance of different system attributes A, B… by rating 1 (equal importance) two attributes which contribute equally to the objective, 9 (absolutely more important) an attribute A which is preferable by the greatest evidence and, symmetrically, 1/9 the attribute B.

Intensity of importance	Definition	Explanation
1	Equal importance	Two factors contribute equally to the objective.
3	Somewhat more important	Experience and judgment slightly favour one over the other.
5	Much more important	Experience and judgment strongly favour one over the other.
7	Very much more important	Experience and judgment very strongly favor one over the other. Its importance is demonstrated in practice.
9	Absolutely more important	The evidence favouring one over the other is of the highest possible validity.
2, 4, 6, 8	Intermediate values	When compromise is needed.

In a pairwise comparison of n elements, for each pair i – j it is possible to determine numerical values of importance aij, for a total of n2 values, of which n(n-1)/2 have to be determined by the analyst, since by definition each value of i and j yields aii = 1 and aij = 1/ aji.

Subsequently, relative weights are calculated to define the relative importance of elements, namely, for example, the values wi e wj such that aij = wi / wj. Therefore, now the problem is to ascertain whether the weights derived by calculation actually reflect the judgment of the analyst.

For this purpose a Consistency Index (CI) is introduced:

$$\mathrm{CI} = \frac{\lambda_{\max} - n}{n-1}$$

where

$$l_{max} * w_i = S_j \, a_{ij} * w_j$$

(CI = 0 indicates perfect consistency, a higher CI means decreasing consistency. Another parameter, the Consistency Ratio (CR), is used to measure how consistent the judgments have been relative to large samples of purely random judgments.

AHP is one of the best known methods of MCDA (Multi-Criteria Decision Analysis), of which dozens of applications exist, on the basis of various categories of mathematical approaches. Actual applications of quantitative techniques with multiple criteria show that this sort of analysis suffers from limitations deriving from an imperfect use of quantitative information available, or from lack of this information.

From a purely mathematical standpoint, selection problems based on multiple criteria do not yield a single solution if the analyst has no further indications on the preferences to be adopted. Methods supporting the selection of the best alternatives follow the objective of finding the alternative which complies to the largest extent with the preferences. Further insight on this approach is reported in the relevant literature.

To complete this review, let’s now briefly consider the Delphi method, which can be defined – as generically as possible – as a method for information collection aiming at a consensus between experts on a specific topic.

In particular, Delphi method can be applied to the formulation of judgments, by expert groups, in cost forecasting processes on the basis of preferences between values. In its traditional formulation, Delphi method is based on a questionnaire which is formulated by a restricted control group and completed individually in an anonymous fashion by a larger group of respondents; for example, this questionnaire could relate to an estimate of most likely cost. After reviewing all responses, the control group communicates them to the respondents, so that they can reformulate their own response as a consequence, having regard to divergent views and the reasons for these views. This procedure can be repeated several times, in order to possibly reach a large convergence of views. In a newer formulation, this process can be accelerated by the use of computer support to process the responses.

Neural Network Techniques

In the last decade, careful consideration was given to the need for improvements in the design cycle of products and systems designed for a long-term utilization. Emphasis was given to reduce design changes, cost and time to market and, as a consequence, life cycle engineering approaches were adopted for a greater effectiveness in the highly competitive global market.

Having noted that 70 percent or more of the Life Cycle Cost of systems is committed by early life cycle decisions, appropriate design techniques had to be devised to meet the aforesaid purpose. Now, it is in the initial phase that the system operational requirements have to be streamlined in order to avoid inconsistencies and drawbacks deriving from lack of detailed design information, and to accelerate the decision making processes. Specifically in the case of SoS, the adoption of parametric models for LCC estimation is complicated by the formulation of a large number of design concepts and requirements to be evaluated in terms of cost (LCC) – effectiveness. Therefore, the more traditional LCC models cannot be applied due to the lack of detailed information required in the initial life cycle phase.

Artificial neural network techniques have been found helpful in approximate LCC evaluations during the Concept Stage. In this type of model, the analyst can compare the properties of a new system with a similar existing system for which LCC studies are available, so that the cost estimation process can be executed on the basis of the properties (usually of higher level) of the new system.

Conclusion

A System of Systems (SoS) is a system of interest whose system elements are themselves systems. Every system in a SoS structure is able to operate in a stand-alone mode, and also to contribute to the achievement of higher-level mission requests. SoS have several specific characteristics that were described. The realization of every SoS involves trade-offs between different solutions and between individual systems’ performance. SoS typically have attributes, including broader technical scope, greater complexity of integration efforts, dynamic and challenging design, re-configurability of system architectures, peculiarity of SoS simulation and modeling, and rigorous interface design and management. Concerning the estimated cost of a SoS, traditional estimation methodologies including Bayes Techniques, Decision Support Methods, and Neural Network Techniques were described.

Plans for Future Research

The economic aspects of System-of-Systems Engineering will be discussed in detail in a forthcoming book concerning Systems Lifecycle Economics, which the author of this article is drafting at this time. The SE Community will potentially benefit from this book in a number ways:

1. Traditionally, in the cost management of complex projects, the most attention has been given to the optimization of system acquisition costs; whereas ownership costs (e.g. operating costs and maintenance costs) are not given the same consideration.

2. Technological innovations characterising advanced systems require a careful balance of the whole acquisition, ownership, and retirement costs, in order to avoid unjustified trends in cost growth.

3. The estimation of Life Cycle Cost, representing the system unit cost over its entire lifetime, offers a number of basic elements to support decisions not only during the early life cycle stages, but in all subsequent periods. Specific decisions are required to manage maintenance policies and to carry out trade-offs between different possible alternatives, until the system life comes to its projected (and possibly, to some extent, unpredicted) end.

Bibliography

B.S. Blanchard, W.J. Fabrycky –"Systems Engineering and Analysis" – Prentice Hall – 2010.
H.A. Linstone, M. Turoff (Eds.) – "The Delphi Method – Techniques and Applications" – New Jersey Institute of Technology – 2002.
G. J. Thuesen, W. J. Fabrycky – "Engineering Economy" –Prentice Hall – 1993.
G. Wang, J.A. Lane, R. Valerdi, B. Boehm – "Towards a Work Breakdown Structure for Net Centric System of Systems Engineering and Management" – 2006.
J.A. Lane, R. Valerdi – "Synthesizing SoS Concepts for Use in Cost Estimation" – 2006.
INCOSE – International Council on Systems Engineering – "Systems Engineering Handbook" – 2007 and further updates.
M. Smit, M. Pica et al. – NATO Publication RTO TR-SAS-054 "Methods and Models for Life Cycle Costing" – June 2007
B.S. Blanchard – "System Engineering Management" – J. Wiley & Sons – 2008
ISO/IEC 15288, "Systems and software engineering – System life cycle processes", 2008.
B. Flynn, M. Pica et al. – NATO Publication RTO TR-SAS-069 "Code of Practice for Life Cycle Costing" – September 2009

Software Quality Metrics:

Three Harmful Metrics and Two Helpful Metrics

June 2012

Part 1 of 2
(Part 2 will be provided in the next issue of SyEN)

Capers Jones

VP and Chief Technology Officer, Namcook Analytics LLC

Abstract

The cost of finding and fixing bugs or defects is the largest single expense element in the history of software. Bug repairs start with requirements and continue through development. After release, bug repairs and related customer support costs continue until the last user signs off. Over a 25 year life expectancy of a large software system in the 10,000 function point size range, almost 50 cents out of every dollar will go to finding and fixing bugs.

Given the fact that bug repairs are the most expensive element in the history of software, it might be expected that these costs would be measured carefully and accurately. They are not. Most companies do not measure defect repair costs, and when they do, they often use metrics that violate standard economic assumptions.

This article discusses three bad metrics and two good metrics. The three bad metrics are: 1) cost per defect; 2) lines of code; and 3) technical debt.

The two good metrics are: 1) function points, for normalization of data; and 2) Defect removal efficiency, for measuring the percentage of bugs found prior to release and afterwards via maintenance.

Introduction

The costs of finding and fixing bugs have been the major cost driver of large software applications since the software industry began. One might think that the software industry would have solid and reliable measurement data on its most expensive activity, but this is not the case.

Many companies do not start to measure bugs or defect repairs costs until testing starts, so all defects and repairs associated with requirements and design are invisible or under-reported.

Even worse, many attempts to measure quality use metrics that violate standard economic assumptions, and conceal or distort the economic value of high quality. There are three very hazardous metrics that all distort quality economics and understate the true value of software quality:

1. Cost per defect;

2. Lines of code for normalization of results; and

3. Technical debt.

All three of these metrics share a common failing. They all ignore fixed costs, which will be dealt with later in this article. Individually, each has other failings too.

There are two helpful and valid metrics that show the economic value of software quality, and also can be used to predict quality and delivered defects, as well as measure:

1. Function points for normalization of results; and

2. Defect removal efficiency.

Let us start by considering the reasons that the three bad metrics are harmful, and then why the two good metrics are useful.

The Errors and Hazards of Cost per Defect

The cost-per-defect metric has been in continuous use since the 1960’s for examining the economic value of software quality. Hundreds of journal articles and scores of books include stock phrases, such as “it costs 100 times as much to fix a defect after release as during early development.”

Typical data for cost per defect varies from study to study but resembles the following pattern circa 2012:

Defects found during requirements = $250
Defects found during design = $500
Defects found during coding and testing = $1,250
Defects found after release = $5,000

While such claims are often true mathematically, there are four hidden problems with cost per defect that are usually not discussed in the software literature and are not well understood:

1. Cost per defect penalizes quality and is always cheapest where the greatest numbers of bugs are found.

2. Cost per defect ignores fixed costs. Even with zero defects there will be costs for inspections, testing, static analysis, and maintenance personnel. These costs are either fixed or inelastic and do not change at the same rate as defect volumes.

3. Because more bugs are found at the beginning of development than at the end, the increase in cost per defect is artificial. Actual time and motion studies of defect repairs show little variance from end to end.

4. Even if calculated correctly, cost per defect does not measure the true economic value of improved software quality. Over and above the costs of finding and fixing bugs, high quality leads to shorter development schedules and overall reductions in development costs. These savings are not included in cost per defect calculations, so the metric understates the true value of quality by several hundred percent.

The cost per defect metric has such serious shortcomings for economic studies of software quality that a case might be made for considering this metric to be a form of professional malpractice for economic analysis of software quality.

Consider a well-known law of manufacturing economics:

“If a manufacturing cycle includes a high proportion of fixed costs and there is a reduction in the number of units produced, the cost per unit will go up.”

Even though every activity is based on fixed and unchanging amounts of time, look at what happens to cost per defect in Table 1:

	Writing Test Cases	Running Test Cases	Repairing Defects	TOTAL COSTS	Number of Defects	$ per Defect
Unit test	$1,250.00	$750.00	$18,937.50	$20,937.50	50	$418.75
Function test	$1,250.00	$750.00	$7,575.00	$9,575.00	20	$478.75

Regression test	$1,250.00	$750.00	$3,787.50	$5,787.50	10	$578.75
Performance test	$1,250.00	$750.00	$1,893.75	$3,893.75	5	$778.75
System test	$1,250.00	$750.00	$1,136.25	$3,136.25	3	$1,045.42
Acceptance test	$1,250.00	$750.00	$378.75	$2,378.75	1	$2,378.75

Table 1: Cost per Defect for Six Forms of Testing
(Assumes $75.75 per staff hour for costs)

As an application moves through a full test cycle that includes unit test, function test, regression test, performance test, system test, and acceptance test, the time required to write test cases and the time required to run test cases stays almost constant; but the number of defects found steadily decreases.

Table 1 shows the approximate costs for the three cost elements of preparation, execution, and repair for the test cycles just cited using a fixed rate $75.75 per hour for all activities:

- Writing test cases takes 16.5 hours for every test stage
- Running test cases takes 9.9 hours for every test stage
- Defect repair takes 5.0 hours for every defect found

What is most interesting about Table 1 is that cost per defect rises steadily as defect volumes come down, even though Table 1 uses a constant value of 5.0 hours to repair defects for every single test stage!

In other words, every defect identified throughout Table 1 had a constant cost of $378.25 and 5 hours when only repairs are considered.

In fact, all three columns use constant values and the only true variable in the example is the number of defects found!

In real life, of course, preparation, execution, and repairs would all be variables. But by making them constant, it is easier to illustrate the main point: cost per defect rises as numbers of defects decline.

Since the main reason that cost per defect goes up as defects decline is due to the fixed costs associated with preparation and execution, it might be thought that those costs could be backed out and leave only defect repairs. Doing this would change the apparent results and minimize the initial errors, but it would introduce three new problems:

1. Removing quality cost elements that may total more than 50% of total quality costs would make it impossible to study quality economics with precision and accuracy.

2. Removing preparation and execution costs would make it impossible to calculate cost of quality (COQ) because the calculations for COQ demand all quality cost elements.

3. Removing preparation and execution costs would make it impossible to compare testing against formal inspections, because inspections do record preparation and execution as well as defect repairs.

Backing out or removing preparation and execution costs would be like going on a low-carb diet and not counting the carbs in pasta and bread, but only counting the carbs in meats and vegetables. The numbers might look good, but the results in real life would not be good.

The bottom line is that cost per defect penalizes quality and makes buggy applications look better than they are because their cost per defect is lower. Cost per defect also makes early defects look cheaper than late defects and has led to the urban legend that “it costs 100 times as much to fix a bug after release than early in development.”

Even worse, the true value of quality is not merely lowering defect repairs, but getting software out earlier, shortening development schedules, lowering maintenance costs, and having happier customers.

Cost per defect has blinded the software industry to the true economic value of software and led to the false assumption that “high quality is expensive.” High quality for software is not expensive, and in fact is much cheaper and faster to develop high quality software than buggy software.

The Errors and Hazards of Lines of Code (LOC)

The “lines of code” or LOC metric has been in continuous use since the 1960’s. Most users of LOC metrics have never studied the behavior of this metric across multiple languages.

As with the cost per defect metric, the LOC of code metric ignores fixed costs. The mathematical result is that low-level languages such as assembly and C seem to be cheaper and of higher quality than modern high-level languages such as Ruby and MySQL.

Let us consider two different languages to see what happens and why LOC metrics are so harmful to economic studies. We will consider the C language and the Java language as examples in Table 2. We will assume that it takes twice as much C code as Java code for a specific application:

	C Language 2000 LOV (2 KLOC)	C Language 1000 LOC (1 KLOC)
Defect Sources	Defects	Defects
Requirements	10	10
Design	20	20
Source Code	30	15
Documents	3	3
Bad fixes	2	2
TOTAL	65	50
Defects per KLOC	32.50	50.00
Defects per Function Point	3.25	2.50

Table 2: Quality Distortion caused by KLOC Metrics

Note that when data is normalized using “defects per KLOC” and all defect sources are included, the lower-level C language has fewer defects per KLOC. This true even though the C version had 65 total defects and the Java version had only 50.

Even if only code defects are considered, there is still a distortion of results with LOC metrics. Code defects for both Java and C are exactly 15 per KLOC even though the C version had twice as many bugs.

LOC metrics have some legitimate uses, but they are not valid for software economic analysis and indeed should be considered to be professional malpractice for that purpose. LOC metrics can be used to examine coding speed, cyclomatic complexity, numbers of test cases, test coverage, and a number of ancillary topics. But LOC metrics are not suitable for economic studies.

The more languages that are included the worse LOC metric become. The following table is from an earlier study that compared 10 languages used for versions of a PBX switching system:

Language	Effort (Months)	Funct. Pt per Staff Month	Work Hrs. per Funct Pt.	LOC per Staff Month	LOC per Staff Hour
Assembly	781.91	1.92	68.81	480	3.38
C	460.69	3.26	40.54	414	3.13
CHILL	392.69	3.82	34.56	401	3.04
PASCAL	357.53	4.20	31.46	382	2.89
PL/I	329.91	4.55	29.03	364	2.76
Ada83	304.13	4.93	26.76	350	2.65
C++	293.91	5.10	25.86	281	2.13
Ada95	269.81	5.56	23.74	272	2.06

Objective C	216.12	6.94	19.02	201	1.52
Smalltalk	194.64	7.71	17.13	162	1.23
Average	360.13	4.17	31.69	366	2.77

Table 3: Productivity Rates for 10 Versions of the Same Software Project
(A PBX Switching system of 1,500 Function Points in Size)

As can be seen, LOC metrics totally reverse real economic productivity and makes the most labor-intensive version using assembly language look faster than the most efficient version that used Smalltalk!

This is a textbook example of LOC as professional malpractice. This table comes from an actual consulting study where developers at a telecommunications company wanted to adopt object-oriented languages but management resisted because their internal LOC data made low-level languages look more productive than high-level languages.

The costs of requirements, design, and other non-coding tasks on modern systems are often more expensive than the code itself. Of the five major cost drivers for software, LOC metrics can only be used for one. The five major cost elements are shown in Table 4.

	Activities	Percent of Costs
1	Finding and fixing bugs	30.00%
2	Coding or programming	25.00%
3	Producing paper documents	20.00%
4	Meetings and communications	15.00%
5	Project management	10.00%
	Total	100.00%

Table 4: Major Software Cost Drivers 2012

LOC metrics have supplemental purposes for software projects, but should never be the primary metric for economic analysis.

The Errors and Hazards of Technical Debt

The concept of technical debt is the newest of the quality metrics, having first been described by Ward Cunningham in a 1992 paper. From that point on, the concept went viral and is now one of the most common quality metrics in the United States and indeed the world.

The essential idea of technical debt is that mistakes and errors made during development that escape into the real world when the software is released will accumulate downstream costs to rectify.

In a sense technical debt tends to piggy back on the "cost per defect" metric with an implied assumption that post-release defects and changes have higher costs than internal defects and changes.

As a metaphor or general concept the idea of technical debt is attractive and appealing. For one thing it makes software quality appear to take on some of the accumulated wisdom of financial operations, although the true financial understanding of the software industry is shockingly naive.

However technical debt suffers from the same problems as cost per defect and lines of code: it ignores fixed costs. It has other and much more serious problems that are not intrinsic, but have come to be unfortunately common.

A major problem with technical debt is that it ignores pre-release defect repairs, which are the major cost driver of almost all software applications. Ignoring pre-release defect repairs is a serious deficiency of technical debt.

Second, what happens after software is released to the outside world is not identical to the way software is developed. You need to support released software with customer support personnel who can handle questions and bug reports. And you also need to have maintenance programmers standing by to fix bugs when they are reported.

This means that even software with zero defects and very happy customers will accumulate post-release maintenance costs that are not accounted for by technical debt. Let us assume you release a commercial software application of 1000 function points or 50,000 lines of Java code.

Prior to release you have trained 2 customer support personnel who are under contract and you have 1 maintenance programmer on your staff assigned to the new application. Thus even with zero defects you will have post-release costs of perhaps $15,000 per month.

After several months you can reassign the maintenance programmer and cut back to 1 customer support person, but the fact remains is that even zero-defect software has post-release costs.

The third and most serious flaw with technical debt concerns the 50% failure rate of large systems > 10,000 function points or 500,000 Java statements in size. If an application of this size is cancelled and not released at all, then technical debt will of course be zero. But your company just wasted $25,000,000 on a project that was terminated due to poor quality!

Yet another omission from the calculations for technical debt are the costs of litigation and punitive damages that might occur if disgruntled clients sue a vendor for poor quality.

Here is an example from an actual case. The shareholders of a major software company sued company management for releasing software of such poor quality that, the shareholders claimed, the poor quality was lowering the stock price.

Clearly the defects themselves would accumulate technical debt, but awards and punitive damages based on litigation are not included in technical debt calculations. In some cases, litigation costs might be high enough to bankrupt a software company.

This kind of situation is not included in the normal calculations for technical debt, but it should be. In other words, if technical debt is going to become a serious concept as is financial debt, then it needs to encompass every form of debt and not just post-release changes. It needs to encompass the high costs of cancelled projects and the even higher costs of losing major litigation for poor quality.

To illustrate that technical debt is only a partial measure of quality costs, Table 5 compares technical debt with cost of quality (COQ). As can be seen, technical debt only encompasses about 13% of the total costs of eliminating defects.

Note also that, while technical debt is shown as $86,141, right above this cost is the much higher cost of $428,625 for pre-release quality and defect repairs. These pre-release costs are not included in technical debt!

Just below technical debt are costs of $138,833 for fixed overhead costs of having support and maintenance people available. These overhead costs will accrue whether maintenance and support personnel are dealing with customer calls, fixing bugs, or just waiting for something to happen. Even with zero-defect software with zero technical debt there will still be overhead costs. These overhead costs are not included in technical debt, but are included in cost of quality (COQ).

	Defects		
Code defect potential	1,904		
Req. & design def. pot.	1,869		
Total Defect Potential	3,773		
Per function point	3.77		
Per KLOC	70.75		
Defect Prevention	**Efficiency**	**Remainder**	**Costs**
JAD	23%	2,924	$37,154
QFD	0%	2,924	$0
Prototype	20%	2,340	$14,941
Models	0%	2,339	$0
Subtotal	38%	2,339	$52,095

Pre-Test Removal	**Efficiency**	**Remainder**	**Costs**
Desk check	25%	1,755	$19,764
Static analysis	55%	790	$20,391
Inspections	0%	790	$0
Subtotal	66%	790	$40,155
Test Removal	**Efficiency**	**Remainder**	**Costs**
Unit	30%	553	$35,249
Function	33%	370	$57,717
Regression	12%	326	$52,794
Component	30%	228	$65,744
Performance	10%	205	$32,569
System	34%	135	$69,523
Acceptance	15%	115	$22,808
Subtotal	85%	115	$336,405
			Costs
PRE-RELEASE COSTS			$428,655
POST-RELEASE REPAIRS (TECHNICAL DEBT)			$86,141
MAINTENANCE OVERHEAD			$138,833
COST OF QUALITY (COQ)			$653,629
Defects delivered	115		
High severity	22		
Security flaws	10		
High severity %		**18.94%**	

Table 5: Technical Debt Compared to Cost of Quality (COQ)
(1000 function points and 50,000 Java statements)

Even worse, if a software application is cancelled before release due to poor quality, it will have zero technical debt costs but a huge cost of quality.

An “average” project of 10,000 function points in size will cost about $20,000,000 to develop and about $5,000,000 to maintain for 5 years. About $3,000,000 of the maintenance costs will be technical debt.

But if a project of the same size is cancelled, they are usually late and over budget at the point of termination, so they might cost $26,000,000 that is totally wasted as a result of poor quality. Yet technical debt would be zero since the application was never released.

The bottom line is that the use of technical debt is an embarrassing revelation that the software industry does not understand basic economics. Cost of quality (COQ) is a better tool for quality economic study than technical debt.

Editor's Note: Part 2 of this article by renowned quality expert Capers Jones will appear in the next issue of the Systems Engineering Newsletter (SyEN).

Systems Engineering News

INCOSE Systems Engineering Certification Discussions

The Board of Directors of the International Council on Systems Engineering (INCOSE) considered at its meeting in Rome earlier this month a number of options for the way forward regarding its systems engineering certification program (Associate Systems Engineering Professional [ASEP]/Certified Systems Engineering Professional [CSEP]/Expert Systems Engineering Professional [ESEP]) that recently delivered the 1000th CSEP. Although there was considerable discussion concerning alternatives, the conclusion reached was that no change to the program should be made at this time.

For more information on the INCOSE Certification Program: http://www.incose.org/educationcareers/certification/

For information on ASEP/CSEP training

ASME Conducts Large Scale Verification and Validation Symposium

ASME (founded as the American Society for Mechanical Engineers) held over May 2 - 4, 2012, in Las Vegas, NV, USA, what it believes to be the first large-scale symposium dedicated entirely to Verification, Validation, and Uncertainty Quantification of computer simulations.

The purpose of the symposium was to bring together engineers and scientists from all disciplines that use computational modeling and simulation to discuss and exchange ideas and methods for verification of codes and solutions, simulation validation, and assessment of uncertainties in mathematical models, computational solutions, and experimental data. The conference included plenary sessions and paper presentation sessions, some of which were oriented by an application field, and some focused by technical goal and approach. The intent was to create an environment where scientists and engineers in various fields, who do not normally have an opportunity to interact, were able to share verification and validation methods, approaches, successes and failures, and ideas for the future.

Preconference activities included ASME V&V standards development committee meetings. ASME V&V Subcommittees currently include: Computational Solid Mechanics, Computational Fluid Dynamics and Heat Transfer, Nuclear Power Thermal/Fluids Systems, and Medical Devices.

The format for this first conference was presentations only, with no requirement for a formal paper submission. A volume of final abstracts is available for registered symposium attendees.

More information

Update to BABOK® Guide Version 3

The International Institute of Business Analysis (IIBA) received over 200 applications in response to its call for volunteers to work on the update of *A Guide to the Business Analysis Body of Knowledge® (BABOK® Guide) Version 3.*

More information

Singapore Hosts the First Business Analysis Congress Asia

For the first time, the Business Analysis community in Asia gathered to discuss the issues, competencies and challenges they are facing today. BAs, Senior BAs and others from Singapore, Malaysia and Indonesia attended the inaugural Business Analysis Congress Asia on May 29 and 30, 2012 in Singapore.

More information

Online Survey on Requirements Engineering for Variability-intensive Systems

The Software Engineering and Architecture Group of the University of Groningen (the Netherlands) is conducting a variability requirements engineering survey with the goal of understanding the needs of practitioners when dealing with variability in the real world. The group defines variability as the ability of a software system or artifact to be adapted (e.g., extended, customized or configured) for use in a specific context. Examples of variability-intensive systems include software product lines or families like MS Office, self-adaptive systems, or open platforms like JEE.

To participate in this survey, the participant should:

- have RE experience (industry, research);
- have experience with industrial software projects (including industry-research collaborations); and
- have an understanding of variability (e.g., from product lines, self adaptive systems, open platforms, anticipating change during RE, or planning different product versions or releases).

To take the survey, which is anonymous and takes only a few minutes: http://www.reviss.org

Featured Society

Design and Technology Teachers' Association Victoria (DATTA Vic.)

DATTA Vic. is a non-profit organization formed in October 1987. From humble beginnings as a group of like minded teachers of technology, DATTA Vic. has grown to a membership of over 650. The DATTA Vic. represents technology teachers in education throughout the State of Victoria, Australia.

You might ask, what does this have to do with systems engineering?

Well, actually, quite a lot. To their huge credit, DATTA Vic. has recognized the significance of systems engineering as an overarching discipline of problem solving applied to technical problems, and has for some time embraced systems engineering within its mantra. Its members teach the Victorian Certificate of Education (VCE) subject “Systems Engineering” to secondary students in their university entrance qualification year. Systems engineering figures predominantly on the Association’s home page.

More information
Course accreditation

INCOSE Technical Operations

AFIS Technical Committees (TCs)

AFIS, Association Française d’Ingénierie Système, acts as the French chapter of the International Council on Systems Engineering (INCOSE). AFIS is a non-profit organization that was founded in 1998. A Memorandum of Agreement defines the terms of an affiliation agreement between AFIS and INCOSE. AFIS is comprised of corporate members (companies, public organizations, Subject Matter Experts (SMEs), Education and Research organizations), and individual members. As of 2012, AFIS included 40 corporate members (22 companies and public organizations, 5 SMEs, and 13 academia), and 450 individual members (including 300 free members from the corporate members). AFIS operates a number of Technical Committees.

Each AFIS Technical Committee (TC) is created by a decision of the AFIS Steering Board. The projects proposed by each Technical Committee are approved and can be funded by the AFIS Steering Board.

A set of two linked books: “*INCOSE SE Hdbk 3.2*” and “*Découvrir et Comprendre l’Ingénierie Système*” are the common reference system for all TCs.

Two books have been developed by an AFIS global effort:

- "*Découvrir et Comprendre l'Ingénierie Système*" (390p). This book is in publication by CEPADUES;
- AFIS White Book called "*Ingénierie Système: La vision AFIS pour les années 2020-2025*".

This latter book will be published by CEPADUES. An English language version is available in PowerPoint format.

All the AFIS documents are in French. English versions are made when necessary (e.g. for international co-operation).

TC SEM (Systems Engineering Management)

Technical area:

For all the systems life cycle phases, from the bid and proposal until disposal:
methodologies, models, and practices, tools used to drive the systems engineering activities in the project management environment.
Systems engineering team organization in a multi-enterprise approach.

Leadership:

Chair: Roland Mazzella, Thales
Co-Chair: Gilles Meuriot, Areva TA

Accomplishments / Products:

- Top ten practices to be applied to succeed a long term relationship between SMEs and industrial companies
- "A practical method to reach consensus and take informed decisions amidst diverse, heterogeneous and conflicting goals"

Current Projects:

- Global owning cost structuring through the system life cycle plans and phases.
- Interface control: Take care for the Product and for the Project
- SE management guidebook for SMEs (in connection with the ISO project: SE for VSMEs)
- SE scoreboard for the technical activities management.

TC SE Global Processes

Technical area:

Improvement of systems engineering processes taking place in all the lifecycle phases:

- SE customization to specific environments
- SE application in specific domains
- Application of crossing-cutting approaches.

Leadership:

Chair: Alain Le Put, MALP Conseil
Co-Chair: TBD

Accomplishments / Products:

- Good practices for requirements engineering
- Good practices for customer needs elicitation
- Engineering requirements and architecture for a product line approach
- SE survey for VSMEs
- Guide to implement a product line approach
- System Requirements in a product line approach
- System Architecture in a product line approach

Current Projects:

- System Modeling approach in a product line approach
- Guidebook to implement LEfSE
- SE for VSMEs : ISO Project based on ISO 29110
- Using CMMI to implement Lean systems engineering

TC Research and Innovation in Systems Engineering (new)

Technical area:

This TC is devoted to academic research with a given level of maturity (TRL 3-5): underpinning theoretical principles, number of technical papers, number of involved researchers or institutions. The technologies well known in other disciplines but whose innovation is to use it in Systems Engineering are also in the technical area for this TC.

Leadership:

Chair: Catherine Devic, EDF
Co-Chair: Bruno Vallespir, University of Bordeaux

TC Human Factors

Technical area:

This TC encompass the scientific understanding of the properties of human capability, the application of this understanding to the design, development, deployment and use of systems and services, and the art of ensuring successful application of Human Factors Engineering on a project.

Leadership:

Chair: Regis Mollard, University of Paris Descartes
Co-Chair: Marion Wolff, University of Paris Descartes

Accomplishments / Products:

- Human Factors seminar, July 2011
- HF chapter in the AFIS White Book SE 2025 Vision

Current Projects:

- Human Factors integration in the SE Life Cycle (why, when, how) leaflet `
- Guidebook: "The role of HF inside systems engineering: towards an integrative approach".

TC Dependability, Maintainability, and Validation of Systems.

Technical area:

All the activities and "ilities" used to build trusted systems: dependability and sustainability, integration, verification and validation, technical and social risks control, all kinds of maintenance: preventive, curative, adaptive and evolutionary are in the scope all of this TC.

Leadership:

Chair: Tony Hutinet Dassault Systems
Co-chair: Eric Levrat, University of Nancy-Lorraine
Co-chair: François Peres, ENI Tarbes

Current Projects:

- I V&V training curricula for SE
- Guidebook: "Good practices to control integration activities"

TC MBSE

Technical area:

Formalized application of modeling to support system requirements, design, analysis, verification and validation activities (beginning in the conceptual design phase and continuing throughout development and later life cycle phases).

This TC is starting up a new orientation with two tracks:

- MBSE with SysML as modeling language. In this case, the objective will be to provide the best enabling means for a better SE with SysML. These activities will reuse and will be complementary to those of the INCOSE MBSE Initiative.
- MBSE with more formal approaches, already successfully applied but underestimated; for instance: DSM (Design Structure Matrix), System Dynamics (Forrester MIT), MATE (Multi Attribute Trade space Exploration). In this case, the objective will be to popularize, make operational, the formal approaches which will have been selected.

Leadership:

Co-chair: Alain Kerbrat, Collesys
Co-chair: Jean Luc Wippler, EADS Cassidian

Current Projects:

- Which modeling techniques to model what?
- Dissemination: MBSE explained to my boss
- Basic constructions catalogue in modeling
- Oldies but goodies: reuse of System Dynamics models

TC Training and Skills

Technical area:

Systems Engineering skills: how to identify and characterize them (universities, enterprises)

Systems Engineering Practices training: students and engineers (continuing training
or on the job training)

Leadership:

Chair: Eric Bonjour, University of Nancy Lorraine
Co-chair: Patricia Pancher, Thales University

Accomplishments / Products:

- SE Student’s competition (RobAFIS) Guide
- AFIS Academia/Industry Forum Guide
- RobAFIS 2011 in Nancy event (Dec 8th 2011)
- BKCASE & GRCSE contribution

Current Projects:

- SE skills reference basement
- RobAFIS 2012 and AFIS Forum 2012
- SE minimum curriculum
- SE training for high schools

TC Systems of Systems and Services, Architecture and Engineering

Technical area:

Best practices dissemination and concept development regarding:

- Systems of Systems engineering
- Services engineering

- Architecture and architecture frameworks
- Complex systems
- Organizations and systems interoperability

Leadership:

Chair: Jean Luc Garnier, Thales
Co-chair: Claude Pourcel, LGPIM

Accomplishments / Products:

- SoS glossary and data model.
- Definitions of services for the benefit of SoS,
- Organizational interoperability (paper)

Current Projects:

- Service Engineering Life Cycles
- System engineering and sustainability

Systems Engineering Tools News

Applying Helping Enterprise Architects Better Relate to TOGAF and DoDAF

Governments in particular are using various frameworks to improve their architectural planning and IT implementation. At the The Open Group Conference in Washington, D.C., July 16-18, Chris Armstrong, President of Armstrong Process Group, examined the use of TOGAF 9 to deliver Department of Defense (DoD) Architecture Framework or DoDAF 2 capabilities. He discussed how to use TOGAF architecture development methods to drive the development and use of DoDAF 2 architectures for delivering new mission and program capabilities. His presentation was also live-streamed free from The Open Group Conference. The discussion now was moderated by Dana Gardner, Principal Analyst at Interarbor Solutions. [Disclosure: The Open Group is a sponsor of BriefingsDirect podcasts.]

More Information

Applying Test Driven Development for Embedded and Real-time Development Using Model Based Testing

Increasing complexity and the need for ever-shorter development cycles are pushing embedded development teams to the limit. To gain a competitive edge, companies are looking toward agile methods, such as Test Driven Development, to help prevent defects by ensuring requirements are understood at the earliest stage and that the right solution is developed. Model-driven development approaches can help teams raise the level of abstraction to improve collaboration and better manage complexity. Linking model-based testing and test driven development offers an opportunity to add real agility to the development process. A webcast looks at how a test-driven development approach can utilize model based testing with the IBM Rational solution for systems and software engineering to deliver complex solutions faster with less risk.

More Information

Gartner Adds No Magic, Inc. to Business Process Analysis Magic Quadrant 2011

No Magic, Inc., a global provider of integrated modeling, simulation and analysis solutions and services, announced that it has been positioned by Gartner, Inc. in the Magic Quadrant for Business Process Analysis Tools.

No Magic's MagicDraw is known in the industry as a object-oriented analysis and design tool. MagicDraw's active validation feature helps ensure enterprise-wide consistency across teams and checks the model against business rules and meta-model constraints, notifying users of any violation.

No Magic was evaluated for the Magic Quadrant for Business Process Analysis Tools based on its ability to execute and completeness of vision. MagicDraw provides strong BPMN, UML, SysML, UPDM, DoDAF, MODAF and SoaML modeling support with simulations using activity execution (OMG UML standard), State Diagram (WC3 SCXML), and full SysML bidirectional parametric execution.

More Information

---

arKItect™ - Seamless Representation of Multi-Scale Systems

According to tool developer Knowledge Inside, arKItect™ provides collaborative graphical modeling of complex systems. A unique model combining different point of views is kept in sync. arKItect™ helps making architecture choices by providing a clear, coherent and structured view of the system.

arKItect™ has been used to define functional software and hardware architecture and to produce specifications and safety documentation.

More Information

IBM and NI Plug Systems Engineering Gap

NI and IBM are building an integration between IBM Rational Quality Manager test management and quality management tool, and NI's VeriStand and TestStand real-time testing and test-automation environment. The integration is designed to plug the gap and provide full traceability of what's defined on the test floor back to design and development, enabling more iterative testing throughout the lifecycle and uncovering errors earlier in the process, well before building costly prototypes. The ability to break down the quality management silos and facilitate earlier collaboration provides an opportunity to reduce costs if you look at the numbers IBM Rational is touting. It is said that a bug that costs $1 to fix on a programmer's desktop costs $100 to fix once it makes its way into a complete program and many thousands of dollars once identified after the software has been deployed in the field. While the integration isn't yet commercialized (it is expected at the end of the third quarter), there is a proof of concept being tested with a half dozen NI/IBM customers.

More Information

Latest Version of Siemens PLM Software's Teamcenter Platform Zeros in on Systems Engineering

Stuart Johnson, Siemens PLM Software's Director of Product Marketing for Teamcenter, advises that Siemens has offered requirement management and other systems engineering functionality as part of the Teamcenter portfolio for some time, but only in standalone modules and components that were not integrated into the core platform's unified data architecture. With Teamcenter 9's release, systems engineering functionality has been brought into the fold -- all the systems definitions of the product and the resulting interconnecting relationships are managed as part of the core product model.

More Information

---

eXpress for Diagnostic Modeling and Analysis from DSI International

eXpress from DSI International, Inc. facilities a diagnostic development process for Life Cycle Cost (LCC) with the aim of optimizing "Design for Testability" and "Design for Diagnosability", to increase maintenance efficiency, safety and operational availability, while reducing cost of ownership.

eXpress, presently in Version 5.12.0, is a fully-featured, off-the-shelf software application providing an environment for the design, capture, integration, evaluation and optimization of system diagnostics, prognostics health management (PHM), and holistic systems testability engineering. Some features include:

- design capture, for both up-front and legacy designs
- diagnostics and testability analysis
- prognostics design influence
- reliability engineering support
- maintainability engineering support
- validation and verification
- sensor optimization and trade-off studies
- 1553 Bus Modeling
- “FMECA Plus”.

More information

STAGE Simulation-based Diagnostic and Prognostic Analysis Tool from DSI International

STAGE from DSI International, Inc. is a simulation-based diagnostic and prognostic analysis tool that provides directly traceability of the simulation to a fully integrated systems diagnostic and prognostics design. STAGE, presently in Version 4, simulates the impact of diagnostic, PHM and ISHM engineering decisions upon the support capabilities of a system during its useful lifetime.

STAGE can be used to evaluate the impact and conduct trade-off analyses upon:

- Operational support
- Condition-Based Maintenance
- Reliability-Centered Maintenance (RCM)
- Mission success
- Remaining useful life
- Diagnostic false alarms
- Costs and sparing

More information

STAGE eDev Technologies, Inc. Accounces inteGREAT Smart Docs

eDev Technologies (www.edevtech.com) announced the initial availability of its requirements authoring technology inteGREAT Smart Docs, a Microsoft Office Word 2010 or 2007 plugin for requirements authoring and collaboration.

"A key challenge faced by organizations today is that business and IT use different tools; businesspeople tend to use tools like Microsoft Word, while IT staff use other technical tools for requirements engineering. This leads to broken communication, rekeying data, redundancy, lack of traceability, and fragmentation of information," says Asif Sharif, Chief Technologist, eDev Technologies.

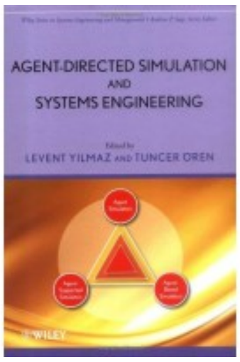
inteGREAT Smart Docs allows business users, managers and analysts to author requirements with the familiar, yet structured, Microsoft Word 2010 or 2007 interface. Information from Smart Docs is automatically saved in the requirements repository, Team Foundation Server, as: goals, risks, market potential, requirements, user stories, actors, events, processes and data; along with their properties, attachments and traceability. "This provides TFS users with a very familiar user interface, will enhance its speed of adoption, while broadening its use throughout the enterprise," says Brian Harry, Product Unit Manager for Team Foundation, Microsoft Corporation.

More Information

Systems Engineering Books, Reports, Articles and Papers

Agent-Directed Simulation and Systems Engineering

Levent Yilmaz and Tuncer Oren



Published by: Wiley Series in Systems Engineering and Management

Publication date

ISBN: 3527407812, 9783527407811

Format: PDF

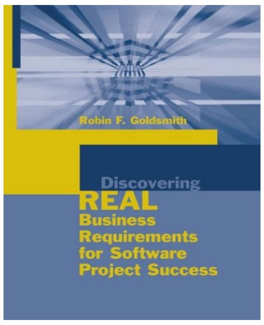
Abstract: The authors provide an overview of the simulation and systems engineering areas; describe principles, methods, tools, and environments; and discuss applications in such areas as testing and evaluation, process performance analysis, decision support, and organization and work system engineering.

“It is probably the only book to date, to present the synergy between modeling and simulation, systems engineering, and agent technologies and to also deal with agent simulation and agent-supported simulation.” (Inside OR, November 2009)

More Information

Discovering Real Business Requirements for Software Project Success

Robin F. Goldsmith



Published by: Artech House

Publication date:

ISBN-13: 978-1580537704

Format: Hardcover

The following review is provided by Johanna Rothman on Amazon.com:

If you ever wanted to know if your requirements were accurate and complete, this book will help. As the author says on p. 13, "...we're going to intermix discovery methods with techniques for testing the adequacy of requirements." Using his problem pyramid, which includes a way to measure the problem and the solution, the author discusses several techniques to elicit and define requirements. He defines 64 techniques to test requirements throughout a project. For example, test method #17, "identify assumptions" is common sense that too few people remember. Test method #53 is "Defining acceptance criteria," a technique useful for any project if you want to know you've built what the customer wanted. Between the problem pyramid emphasis on measurement and the various tests, the book can help you meet its promise of preventing too much requirements change throughout a project.

New Whitepaper from Vitech Corporation Guides Systems Engineers Seeking Most Effective Design

Vitech Corporation is now offering free access to its new 19-page whitepaper entitled *9 Laws of Effective Systems Engineering*, authored by Zane Scott, Vitech's vice president of professional services and author of the recently released book, *A Primer for Model-Based Systems Engineering*. Vitech is making the whitepaper available to systems engineers, project designers and managers, engineering students, or anyone interested in learning more about model-based systems engineering.

More Information

Systems Journals

The following is a list of journals on systems in general (as distinct from journals specifically on systems engineering):

Acta Cybernetica
http://www.inf.u-szeged.hu/actacybernetica/starten.xml

Complex Systems
http://www.complex-systems.com/

Constructivist Foundations (e-journal)
http://www.univie.ac.at/constructivism/journal/

Cybernetics and Systems: An International Journal
Taylor & Francis: http://www.tandfonline.com/toc/ucbs20/current

Cybernetics and Human Knowing
Imprint Academic: http://www.imprint-academic.com/C&HK

Dynamical Systems: An international journal
Taylor & Francis: http://www.tandfonline.com/toc/cdss20/current

Emergence: Complexity and Organization [E:CO]
http://www.emergence.org/

Entropy
MPDI: http://www.mdpi.com/journal/entropy

European Journal of Control
Hermes Science: http://journals.dei.polimi.it/ejc/

European Journal of Economic and Social Systems (in French)
Lavoisier: http://www.lavoisier.fr/gb/e-revues/index.asp?texte=0947-3580&select=issn&exact=on&from=RevuesOnline

European Journal of Information Systems
The OR Society/Palgrave Macmillan: http://www.palgrave-journals.com/ejis/index.html

European Journal of Operational Research
Elsevier: http://www.journals.elsevier.com/european-journal-of-operational-research/#description

IEEE Transactions on Systems, Man, and Cybernetics
IEEE: http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=3477

Interdisciplinary Description of Complex Systems
http://www.indecs.eu/

International Journal of Control
Taylor & Francis: http://www.tandfonline.com/toc/tcon20/current

International Journal of Human-Computer Studies
Elsevier: http://www.sciencedirect.com/science/journal/10715819

International Journal of Man-Machine Studies (now renamed)
Elsevier: http://www.sciencedirect.com/science/journal/00207373

International Journal of General Systems
Taylor & Francis: http://www.tandfonline.com/toc/ggen20/current

International Journal of Information Technologies and Systems Approach
IRMA/IGI Global: http://www.igi-global.com/journal/international-journal-information-technologies-systems/1098

International Journal of Systems Science
Taylor & Francis: http://www.tandfonline.com/toc/tsys20/current

Journal of Applied Systems Analysis (Now published as the European Journal of Information Systems)
Back copies: http://www.periodicals.com/html/ihp_e.html?ej50581

Journal of Applied Systems Studies
Cambridge: http://www.unipi.gr/jass/

Journal of Organisational Transformation and Social Change
Intellect: http://www.intellectbooks.co.uk/journals/view-journal,id=128/

Journal of the Operational Research Society
The OR Society/Palgrave: http://www.palgrave-journals.com/jors/index.html

Journal of Complexity
Elsevier: http://www.journals.elsevier.com/journal-of-complexity/

Journal of Social and Evolutionary Systems (until 1999 known as Journal of Social and Biological Systems)
Elsevier: http://www.sciencedirect.com/science/journal/10617361?oldURL=y

Journal of Systems Management (no longer published)
Back issues: http://www.accessmylibrary.com/archive/412190-journal-of-systems-management.html

Journal of Systems Science and Complexity
Springer: http://www.springer.com/mathematics/applications/journal/11424

Journal of Information Science
Sage: http://jis.sagepub.com/

Kybernetes
Emerald: http://www.emeraldinsight.com/products/journals/journals.htm?id=k

Open Systems and Information Dynamics
Springer: http://www.springer.com/physics/complexity/journal/11080

SIAM Journal on Applied Dynamical Systems
SIAM: http://www.siam.org/journals/siads.php

Simulation
SCS/Sage: http://www.sagepub.com/journalsProdDesc.nav?prodId=Journal201571

Systemist
UKSS: http://www.first-pages.com/ukss/publications/systemist

Systems Dynamics Review
SDS/Wiley: http://onlinelibrary.wiley.com/journal/10.1002/%28ISSN%291099-1727

Systems Research and Behavioral Science
IFSR/Wiley: http://onlinelibrary.wiley.com/journal/10.1002/%28ISSN%291099-1743a

Systemic Practice and Action Research
Springer: http://www.springer.com/business+%26+management/journal/11213

Conferences and Meetings

International Conference of the System Dynamics Society, 2012
July 22 - 26, 2012, St. Gallen, Switzerland
More Information

ASME 2012 International Design Engineering Technical Conferences (IDETC) and Computers and Information in Engineering Conference (CIE)
August 12 - 15, 2012, Chicago, IL, USA
More information

4th Improving Systems & Software Engineering Conference (ISSEC) 2012
August 15 - 16, 2012, Melbourne, Australia
More information

KSE 2012
August 17 - 19, 2012, Danang, Vietnam
More information

EmpiRE 2012 : Workshop on Empirical Requirements Engineering
August 25, 2012, Chicago, USA
More information

Workshop on Model-Driven Engineering for Networked Ambient System (MDE4NAS)
August 27 - 29, 2012, Niagara Falls, Canada
More information

9th INCOSE SA Conference – The Jewel in the Crown NEW
August 29 – 27, 2012, Pretoria, South Africa
More information

18th International Symposium on Formal Methods
August 27 - 31, 2012, CNAM, Paris, France
More information

Summer School 2012: Verification Technology, Systems & Applications NEW
September 3 – 7, 2012, Saarbrücken, Germany
More information

3rd International Summer School on Domain Specific Modeling - Theory and Practice
September 10 - 14, 2012, Lisbon,
More information

Sixth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2012)
September 10 - 14, 2012, Lyon, France
More information

International Workshop on Enterprise Integration, Interoperability and Networking (EI2N'2012) NEW
September 12 - 13, 2012, Rome, Italy
More information

10th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2012)
September 18 - 20, 2012, London, United Kingdom
More information

12th International Workshop on Automated Verification of Critical Systems (AVoCS 2012)
September 18 - 20, 2012, Otto-Friedrich University in Bamberg, Germany
More information

2012 Interdisciplinary Symposium on Complex Systems
September 19 - 24, 2012, Kos island, Greece
More Information

Risk Engineering Society Conference (RISK 2012)
September 20 – 22, 2012, Sydney, Australia
More information

Verifikation a validierung Herausforderungen Bei Kurzen Entwicklungszeiten (V&V Forum) NEW
September 21, 2012, Frankfurt, Germany
More information

RePa 2012 : Second International Workshop on Requirements Patterns
September 24, 2012, Chicago, USA
More information

20th International Requirements Engineering Conference (RE 2012)
September 24 - 28, 2012, Chicago, Illinois, USA
More information

SAFECOMP 2012
September 25 – 28, 2012, Magdeburg, Germany
More information

2nd Requirements Symposium
September 27, 2012, Berlin

MODELS 2012, ACM/IEEE 15th International Conference on Model-Driven Engineering Language & Systems
September 30 - October 5, 2012 – Innsbruck, Austria
More Information

SAM Workshop 2012 NEW
October 1 – 2, 2012, Innsbruck, Austria
More information

6th INCOSE Annual Great Lakes Regional Conference 2012
October 12 – 13, 2012, Schaumburg, Illinois, U.S.A
More information

World Engineering Education Forum (WEEF12)
October 15 - 18, 2012, Buenos Aires, Argentina
More information

19th Working Conference on Reverse Engineering
October 15 - 18, 2012, Kingston, Ontario, Canada
More information

ASME 2012 Dynamic Systems and Control Conference (DSCC2012)
October 16 - 20, 2012, Ft. Lauderdale FL , USA
More information

Human Factors and Ergonomics Society HFES 2012 Annual Meeting
October 22 - 26, 2012, Boston, MA, USA
More information

ICSSEA 2012 NEW
October 23 - 25, 2012, Paris, France
More information

The World Congress on Engineering and Computer Science 2012
October 24 - 26, 2012, San Francisco, USA

8º Congresso Brasileiro de Sistemas NEW
October 25 - 26, 2012, campus da PUC Minas em Poços de Caldas, MG, Brasil
More information

Building Business Capabilities (BBC) 2012
October 28 - November 2, 2012, Fort Lauderdale, FL, USA
More information

12th Annual CMMI Technology Conference and User Group
November 5 – 8, 2012, Denver, USA
More information

INCOSE UK Annual Systems Engineering Conference 2012
November 7 - 8, 2012, London, UK
More information

Systems Engineering Day 2012 (TdSE 2012)
November 7 - 9, 2012, Paderborn, Heinz Nixdorf Museums Forum, Germany.
More information

14th International Conference on Formal Engineering Methods (ICFEM 2012)
November 12 - 16, 2012, Kyoto Research Park, Kyoto, Japan
More information

3rd International Conference on Complex Systems Design & Management (CSD&M 2012)
December 12 - 14, 2012, Cité Internationale Universitaire, Paris, France
More information

INCOSE International Workshop IW2013
January 26 - 29, 2013, Jacksonville, FL, USA
More information

Conference Digital Enterprise Design & Management (DED&M 2013)
February 11 - 12, 2013, Paris, France
More information

International Symposium on Engineering Secure Software and Systems (ESSoS) NEW
February 27 – March 3, 2013, Paris, France
More information

INCOSE IL 2013 NEW
March 5 – 6, 2013, Daniel Hotel Herzlia
More information

The Requirements Engineering Track - 6th Edition at The 28th Annual ACM Symposium on Applied Computing (SAC 2013)
March 18 - 22, 2013, Coimbra, Portugal
More information

11th Annual Conference on Systems Engineering Research (CSER 2013)
March 19 – 22, 2013, Atlanta, Georgia, USA
More information

SysCon 2013
April 15 - 18, 2013, Orlando, FL, USA
More information

SETE 2013 NEW
April 29 – May 1, 2013, Canberra, ACT, Australia

12th International Symposium of the Analytic Hierarchy Process/Analytic Network Process (ISAHP 2013)
June 23 – 26, 2013, Kuala Lumpur, Malaysia
More information

IS 2013 – Philadelphia NEW
June 24 – 27, 2013, Philadelphia, Pennyslvania USA
More information

ASME 2013 International Design Engineering Technical Conference and Computers and Information in Engineering Conference (IDETC/CIE2013)
August 4 - 7, 2013, location TBA, USA
More information

APCOSE 2013 NEW
September 9 - 11, 2013, Keio University in Japan
More information

19th World Congress of the International Federation of Automatic Control (IFAC 2014)
August 24 - 29, 2014, Cape Town, South Africa
More information

9th European Systems Engineering Conference (EuSEc 2014) - INCOSE EMEA 2014 Sector Conference NEW
October 2014, Cape Town, South Africa

Education and Academia

Associate Professor Position Openings in Computer Science and Engineering at Politecnico di Milano, Italy

The Research Area of Computer Science and Engineering at the Dipartimento di Elettronica e Informazione (DEI) of Politecnico di Milano is considering top candidates from foreign institutions at the associate professor level, in order to open new positions. Candidates should either already hold an associate professor position, to use the option of transferring their position to DEI, or have a very strong curriculum in Computer Science and Engineering, in which case they will be invited to participate in a competition which will be offered according to the national and local regulations during 2013.

For a first, informal contact, please send mail to prof. Letizia Tanca (tanca (at) elet.polimi.it), head of the Computer Science and Engineering Section, and to Laura Caldirola, the Section secretary (caldirola (at) elet.polimi.it), preferably by July 31, 2012.

More information

The University of Texas Creates a New Department of Systems Engineering

The University of Texas, Dallas (USA) Erik Jonsson School of Engineering and Computer Science has created a new Department of Systems Engineering. “This is the next step in fulfilling the Jonsson School’s strategic plan of becoming one of the great research engineering schools in the world,” said Dr. Mark W. Spong, Dean of the Jonsson School since 2008 and holder of the Lars Magnus Ericsson Chair in Electrical Engineering and the Excellence in Education Chair. UT Dallas leaders say that the new department leverages two of the University’s key strongholds: engineering and management.

More Information

Systems Engineering Taught in Australia at Secondary School Level

Systems engineering has been taught as a subject at secondary school level (typical student ages 17-18 years) since 2007, as a Victorian Certificate of Education (VCE) subject contributing to University admission, in the State of Victoria, Australia.

In the words of the Victorian Curriculum and Assessment Authority about the accredited "Systems Engineering" VCE subject: "Contemporary society is exposed to the rapid advancement and pervasive influences of technology. Technological systems play an increasingly significant role in the human world. They mediate or control many aspects of human experience. Systems Engineering provides an opportunity for students to develop capabilities in, and knowledge about, the design, operation, construction, assembly, maintenance, diagnosis, repair and evaluation of technological systems, applicable to a diverse range of fields such as engineering, manufacturing, automation, control technologies, mechatronics, electrotechnology, robotics, and energy management. Students gain awareness and understanding of the interactions of these systems with human society and natural ecosystems."

The study promotes innovative thinking and problem-solving skills through a project-based learning approach. It provides opportunities for students to learn about and engage with systems from a practical and purposeful perspective. The study emphasises integration of basic engineering and physics theory with practical tasks. Technological principles and the associated mathematics are incorporated as essential tools employed in the processes of technological systems design, modification, production and evaluation.

The Systems Engineering subject was reviewed in 2011 for implementation in 2013, covering the years 2013-17. A Study Design, Resources and Study Summary, all intended for teacher use, are downloadable at http://www.vcaa.vic.edu.au/vce/studies/futuresd.html.

More information

Some Systems Engineering-Relevant Websites

http://maintenanceforums.com/eve/forums/a/tpc/f/209103451/m/684101634

This page has a detailed and informed discussion of the difference between Failure Modes and Effects Analysis (FMEA) and Failure Modes, Effects and Criticality Analysis (FMECA)

http://maintenanceforums.com/eve/forums

This is an informative forum site covering reliability engineering, maintainability engineering, maintenance, machinery condition monitoring and predictive maintenance, Enterprise Asset Management (EAM) & Computerized Maintenance Management Systems (CMMS)

http://www.weibull.com/

With over 6,000 pages, weibull.com is a vast website devoted entirely to the topic of reliability engineering, reliability theory and reliability data analysis and modeling. The site includes sections on important reliability engineering disciplines, including but not limited to: Reliability Life Data Analysis (Weibull Analysis), Accelerated Life Testing, System Reliability and Maintainability Modeling and Analysis, Reliability Growth Analysis, FMEA & FMECA, Reliability Centered Maintenance (RCM) and Design of Experiments (DOE).

http://users.aber.ac.uk/cwl/SWFMEA/PAPERS/position/position.html

This 1996 site contains a position paper by Chris Loftus, David Pugh and Ian Pyle on Software Failure Modes, Effects and Criticality Analysis.

Standards and Guides

Application Guidance on
ISO/IEC 15288 (IEEE Std 15288-2008) - 2nd Edition 2008-02-01
Systems and Software Engineering - System Life Cycle Process

by Robert Halligan, FIE Aust

1. Background

ISO/IEC 15288:2008 is a process standard of considerable significance, intended to help:

- an organization establish an environment of desired processes;
- a project select, structure and employ the elements of an established environment to provide products and services; and

- an acquirer and a supplier develop an agreement concerning processes and activities. Via the agreement, the processes and activities in the International Standard are selected, negotiated, agreed to and performed.

ISO/IEC 15288:2008 is also intended for use by process assessors — to serve as a process reference model for use in the performance of process assessments that may be used to support organizational process improvement.

The experience of the author is that process standards do not necessarily always achieve the objectives established for them. Years of participation in ISO/IEC and other standards development efforts in the field of systems engineering leads the author to conclude that process standards almost always represent the lowest common denominator of agreement amongst participants. Further, process standards are often developed in highly political environments replete with political agenda. As a consequence, published standards may be less than perfect!

The purpose of this paper is to provide information for consideration by any user, or potential user, of ISO/IEC 15288:2008, with a view to maximizing value that can be achieved in relation to ISO/IEC 15288:2008 practices, and minimising any loss that could arise from use of ISO/IEC 15288:2008 practices.

2. ISO/IEC 15288:2008 Application Guidance

Application guidance is provided in tabular form, keyed to the paragraph numbers of ISO/IEC 15288:2008. Issues are rated in severity from 1 (lover severity) to 10 (high severity).

15288 Para	Application Guidance
1.1 Scope	1.1 Scope The standard states that it may be applied at any level in the hierarchy of a system's structure, and may be configured with hardware, but then goes on to say "when the system element is hardware, refer to other international standards outside the scope of SC7". The statements appear to be contradictory. Apart from unitary elements that have no internal structure, e.g. a conventional coin formed from a material, hardware elements (including information technology) are invariably systems, in accordance with 15288 and dictionary definitions. The exclusion of purely hardware elements from coverage of the standard, if intended, is unfortunate, as some of the most effective implementations of systems engineering that I have found have been in companies who engineer hardware products, and use systems engineering as a major tool for achieving customer satisfaction and commercial success. Severity: 8
4.9 enabling system	The definition in ISO/IEC 15288:2008 of an "enabling system" admits the possibility of an "enabling system" contributing directly to the function of the system-of-interest, and therefore being a part of the system of interest. The verb "supports" admits any system which interoperates with the system-of-interest. For these reasons, the definition is inappropriate. The definition is at odds with the concepts of concurrent engineering (also known as simultaneous engineering), a concept which the concept of enabling systems is fundamentally intended to support. Only the example in ISO/IEC 15288:2008 of a production system as an enabling system helps correct the impression conveyed by the definition. A more suitable definition of an enabling system is "a system which enables one or more phases of the life-cycle of the system-of-interest, whilst not itself being a part of the system-of-interest" NOTE: The consequence of a system-of-interest/enabling system relationship is that the internal design of one depends on the internal design of the other, often leading to the practice of concurrent engineering. Severity: 6
4.22 qualification	The definition of qualification fails to differentiate between the management action of deeming an item qualified, usually for a defined purpose, and the technical activity of verifying that an item complies with the requirements which apply to that item. The ISO/IEC 15288:2008 definition of qualification is inconsistent with the Oxford English Dictionary (OED) and therefore in conflict with the ISO Directives (the rules for developing ISO standards). The distinction between qualification and verification is very, very useful in managing technical projects, but is lost in ISO/IEC 15288:2008. Severity: 4
4.27 security	The inclusion in ISO/IEC 15288:2008 of reliability within the definition of security is highly unconventional and not supported by the OED. The disciplines of reliability engineering and security engineering are substantially different as to knowledge base and methods.

	Severity: 2
4.31 system	The definition in ISO/IEC 15288:2008 of "system" is in fact a definition of an engineered system, not a definition of a system in general. The definition is flawed as a definition of an engineered system by limiting to "for a stated purpose". Is a system engineered to a known but unstated purpose not a system? Severity: 1
4.34 task	This is an obtuse definition not well supported by OED. Severity: 1
4.37 validation	The definition of validation, by invoking the flawed ISO 9000:2005 definition, creates the same problem that exists with ISO 9000:2005 and ISO 9001:2008. Validation is a sub-category of verification under this ISO 9000:2005 definition. The definition departs from the vastly more widely used and highly beneficial distinction between verification and validation: verification – does the item comply with the requirements for the item (OED definition of requirement); validation – does the item satisfy the need for the item (OED definition of need). This distinction between verification and validation comes about because of the reality that requirements can be wrong, and requirements are inevitably incomplete. Thus, it is possible to satisfy requirements, but fail to satisfy the need. To develop successful systems, we must be concerned with both meeting requirements and satisfying need. Severity: 10
5.1.2 Systems	The reference to "its architecture and its system elements" is confusing and lacks logic, since the identification of the system elements is a part of the (physical) architecture of a system. Severity: 2
6.1.1.1 Purpose (of Acquisition Process)	The purpose of the Acquisition Process is not only to obtain a product or service in accordance with requirements (imperatives); it is also to acquire the most overall effective (best) amongst solution alternatives. Severity: 6
6.1.1.3 a) 2)	Goals and value relationships need also to be included, where goals exist. Severity: 6
6.1.2.3 a)	The heading of 6.1.2.3 a) is incorrect. The supplier responds to a request for tender (or request for proposal) with a tender. The acquirer responds to the tender (proposal) by accepting it, rejecting it, or negotiating change. Severity: 3
6.1.2.3 b)	There are many circumstances where it will be entirely appropriate for a supplier to prepare a response that does not satisfy the solicitation, e.g. where some requirements in the solicitation are infeasible or conflicting, or where the supplier's interests are best served by making a counter-offer. A supplier, to be able to claim compliance with ISO/IEC 15288:2008, must only submit fully compliant tenders. That is absurd. Severity: 8
6.2.2.1 Purpose (of Infrastructure Management Process)	This paragraph needs to emphasize the status of the infrastructure elements as elements of one or more enabling systems (e.g. Project System, Production System, Maintenance System), the elements needing to form part of an optimum design of each respective enabling system, complementing and in balance with the human elements of the enabling system, and subject in their development to the practice of concurrent engineering. This critically important relationship is ignored, except for an oblique reference in 6.2.2.3 b) 2). Severity: 6
6.2.4.1 Purpose (of Human	This paragraph needs to emphasize the status of human resources as elements of one or more enabling systems (e.g. Project System, Production System, Maintenance System), needing to form part of an optimum design of each enabling system, complementing and in balance with the infrastructure elements of the enabling system, and subject in their development (e.g. by training) to the practice of concurrent engineering. This critically important relationship

Resource Management Process)	is ignored. Severity: 7
6.2.5.3 b) 2)	Basing quality objectives on stakeholder requirements alone relies on those requirements being consistent with, and an adequate statement of, stakeholder needs. This is rarely the case. Quality objectives must be based on stakeholder requirements, values and needs, to best serve the stakeholders.
6.3.1.1 Purpose (of Project Planning Process)	The status of the Project System as an enabling system, subject in its development to the practice of concurrent engineering with respect to the system(s) to be engineered, needs to be emphasized. This critically important relationship is ignored. Many a project has been seriously compromised by plans and technical realities/decisions never having been aligned, or by becoming misaligned. Severity: 7
6.3.1.3 c) 1)	In requiring a plan for technical management and execution of the project, as opposed to one or a set of plans, ISO/IEC 15288:2008 rules out the use of empowered Integrated Product Teams (IPTs), that have been so successful in shortening timeframes, reducing cost, and increasing product quality. An empowered IPT will do its own planning, within enterprise-wide and project wide constraints, and consistent with higher level planning which generates the tasking of the IPT. Severity: 7
6.3.3 Decision Management Process	Placing decision management as a Project process can convey the impression that technical decisions can and should be referred to, and made by, those in a project management role. This is at odds with the principles of IPTs, which have been enormously successful. Real IPTs are empowered to make decisions within their bounds of assigned responsibility for realization of a system or a system element. This Decision Management process needs to link strongly into the technical processes that involve decision making. Severity: 5
6.3.4 Risk Management Process	Notwithstanding its ISO/IEC 16085-AS 4360 heritage, this whole section of the standard is poor indeed, primitive, missing the point regarding risk, and regarding effective risk management: • risk is expected loss, not a thing that could go wrong (that's a threat). • risk is with reference to a defined level of valued outcome, e.g. cost, schedule, capability, safety, national security, social benefit, political outcome, etc. • risk has the ingredients of the value of the outcome, what can go wrong that threatens that outcome, and how vulnerable we are if that threat is realized. When these ingredients and their probabilities are convolved, we end up with a relative probability of different degrees of loss with respect to a valued outcome, due to a set of threats relevant to that outcome. In a major project, there are usually thousands of threats contributing to the level of risk with respect to a valued outcome (e.g. contributing to the cost risk). • effective risk management begins with an understanding of risk! • effective risk management then involves ensuring that uncertainties (leading to risk and opportunity) are factored into all project decisions, regardless of who makes them • effective risk management relies on people who are making decisions, doing so on an expected value basis (value, taking into account balance of probabilities), either informally, or for important decisions, formally. Severity: 9
6.4.1.1 Purpose (of Stakeholder Requirements Definition Process)	The statement of purpose needs to include the resolution of conflicts between stakeholder requirements, especially conflicts between the requirements of different stakeholders. Severity: 4
6.4.1.2 Outcomes	The statement "Stakeholder requirements for validation are identified" is ambiguous and incomplete. The statement should read "Stakeholder requirements for system verification and system validation are identified and specified"

	Severity: 5
6.4.1.2 Outcomes	Stakeholder measures of effective, goals and value relationships are also an important outcome. Without this information, developers of solutions have no sound basis for selecting among feasible solution alternatives, and no basis for solution optimization. Severity: 7
6.4.1.3 Activities and Tasks	The section needs to recognize and reflect primary stakeholders (the stakeholders that the supplier is serving – e.g. their company or shareholders), versus secondary stakeholders (the stakeholders who are not primary stakeholders, but whose interests influence the interests of the primary stakeholders – e.g. customers, operational users). Otherwise, once MOEs and goals are recognized, the standard can have the effect of requiring the supplier to act as a charity. To fail to acknowledge and deal with these realities considerably reduces the real-world relevance of ISO/IEC 15288:2008. Severity: 6
6.4.1.3 a) NOTE	The statement “Stakeholder requirements describe the needs, wants, desires, expectations and perceived constraints of identified stakeholders” is very damaging to ISO/IEC 15288:2008, because it is totally at odds with the English language as defined by the Oxford English Dictionary. As a result, the statement violates the ISO rules for developing ISO standards. Requirement: an order, a demand, an imperative (OED) Need: a condition of lacking or acquiring some necessary thing, either physically or psychologically (OED) Want: wish for possession of (OED) Expectation: an instance of expecting or looking forward (OED) Constraint: A limitation or restriction (OED). Severity: 9
6.4.1.3 a)	This section has many points of language, detail, and incompleteness which diminish its value. Severity: 4
6.4.2 Requirements Analysis Process	If the “system” referred to in 6.4.2 is the same system as referred to in 6.4.1, then the “Requirements Analysis Process” is a duplication of 6.4.1.3 b) and c), now saying, in effect, “having done 6.4.1.3 b) and c) poorly, now do 6.4.1.3 b) and c) properly. This section refers to a “technical view”, “from the supplier’s perspective”. “Technical” means “relating to technology” or “relating to technique”. So in what sense it the view “technical”? This process has the supplier telling the acquirer what the acquirer’s requirements are, creating the opportunity for the supplier to manipulate the agreement to supply what the supplier wants to supply, reflecting the worst aspects of non-performing acquisition systems worldwide. Yes, 6.4.2 refers to satisfaction of stakeholder requirements, but with stakeholder requirements made purposefully vague under 6.4.1 (implicitly not measurable, since the Requirements Analysis Process is to result “in measurable system requirements”. The Requirements Analysis Process is conspicuous in its absence of reference to satisfaction of stakeholder needs. It is not altogether clear that the system referred to in 6.4.2 is the same system as referred to 6.4.1. Stakeholders generally seek to achieve outcomes. End-use items, engineering systems, production systems, transition systems, maintenance systems and disposal systems (for example) are all a part of the means of doing so, i.e. are a part of solution. The term “product solution”, referred to in 6.4.2.2 but not in 6.4.1, is commonly used to refer to an end-use product alone. If the references to “system” in 6.4.1 and 6.4.2 are not intended to be references to the same system, then 6.4.2 is mandating a “then a miracle occurs” process, as did its predecessor, ISO/IEC 15288:2002. This latter view of the world has been one of the three primary contributors to most failures of large projects. Severity: 10
6.4.2.3 a) 1)	This content essentially duplicates 6.4.1.4 b) – if the same system is being referred to. Severity: 10
6.4.2.3 a) 1)	“Mass” is neither an interface constraint nor is it involved in defining the functional boundary of a system. Mechanical and thermal flows may be involved in defining requirements other than behavioural requirements of the system at its boundary.

	Severity: 3
6.4.2.3 a) 2)	2) overlaps in content substantially with 1). Sub-para 2) is a much more sound statement of good practice in requirements analysis than is 1), however, 1) contains some valid additional content. Severity: 3
6.4.2.3 a) 3)	“Unavoidable solution limitations” not introduced in (valid) stakeholder requirements are not system requirements, just because they are unavoidable – or thought to be so. This requirement of ISO/IEC 15288:2008 violates the principle of maintaining a clear distinction between problem and solution, destroying any basis for design and system verification, and leaving the designer in ignorance of what can be changed in a design, and what cannot be changed in design, if the system requirements change. Severity: 6
6.4.2.3 a) 4)	The note is misleading. Critical performance measures may or may not be measures of effectiveness. In the former case (where more or less with respect to some measure is better), critical performance measures are not associated with measures of effectiveness, they are measures of effectiveness. This note gives weight to the theory that different systems are being referred to in 6.4.1 and 6.4.2. If that is the case, the critical performance measures are derived measures of effectiveness of one or more sub-systems. The note also seems to be mixing up what are conventionally called Technical Performance Measures (TPMs), used to instrument a project primarily for purposes of risk management, with Measures of Effectiveness, used to evaluate and select between feasible design alternatives, and for design optimization. Severity: 7
6.4.2.3 a) 5)	The paragraph says “specify system requirements and functions”, implying that system requirements and functions are somehow mutually exclusive. Where does that leave functional requirements? Severity: 4
6.4.2.3 a) 5)	The note seems to muddle up problem and solution. The principle of maintaining a clear distinction between problem and solution applies no less to safety requirements and safety solution, and similarly security, for the same reasons as previously stated. Severity: 5
6.4.2.3 b) 2)	This requirement confirms the suspicion that the standard is advocating that the supplier, in performing requirements analysis, invent requirements information, then seek stakeholder acceptance of the inventions. The standard should, in fact, be advocating the identification in requirements analysis of each requirements issue, followed by dialogue with the relevant stakeholder(s) on that issue, to resolve the issue. If the supplier is delegated responsibility by the acquirer to invent requirements on behalf of the acquirer (which usually involves a conflict of interest in commercial transactions), the standard should be advocating the use of design processes, not requirements analysis processes, for doing so. Severity: 9
6.4.3 Architectural Design Process	Calling this process the “Architectural Design Process” conflicts with the definition of architecture in the standard, and in OED, and in use throughout engineering. The naming is an unfortunate throw-back to ISO/IEC 15288:2002, which defined “architecture” as design at high physical levels, as contrasted with design at low physical levels. Meanwhile, the rest of the world has used the word “architecture” to mean design at a conceptual level of detail, as distinct from an implementable level of detail. The process should be read as, and called, “Design Process”. Subsequent comments are predicated on this interpretation. Severity: 9
6.4.3.2 f)	The standard needs to distinguish between building the system in system integration, and building the system in production. For some systems, these builds are identical, e.g. a one-off air traffic management system. For other systems, the two build structures can be very different, e.g. a commercial aircraft. The Design Process is concerned with establishing a basis for both system integration builds and production builds. The latter necessitates concurrent engineering practices, which in turn impact on the system integration build structure (or should do so).

	Severity: 5
6.4.3.3 a) Activities and Tasks, Define the Architecture	6.4.3.3 a) is missing the fundamental activity of conceptualization of physical solution alternatives, perpetuating the misunderstandings and misinformation in IEEE 1220 that has done so much damage to enterprises that have tried to follow that standard. The relationship between logical and physical (structural) design – two sides of the same coin - is missing. The words in 2) “Partition the system functions identified in requirements analysis … Generate derived requirements as needed for the allocations.” and the words of 1) mean exactly the same thing, placing in question the credibility of this standard. Severity: 10
6.4.3.3 a)	The words "partitioning" and "allocating" are used loosely. Severity: 3
6.4.3.3 a)	This section talks about “the architecture”, rather than the divergent process of conceptualizing design alternatives, and the convergent process of making decisions between design alternatives, progressively from high levels of abstraction to implementable level of abstraction (i.e. from architecture to detailed design of the system-of-interest with reference to a physical level one level below the system of interest) Severity: 8
6.4.3.3 a)	This section totally ignores the role of non-functional requirements in framing alternative architectures and detailed designs. Severity: 8
6.4.3.3 b) 1)	This paragraph and the related note are without meaning. The “design criteria” for each element were defined in the activity of 6.4.3.3 a) Severity: 4
6.4.3.3 b) 2)	This paragraph is spurious. The activity has already been performed in the activity of 6.4.3.3 a) – or should have been. Also, the paragraph seems to get into trouble with the distinction between use, and operation by an operator. Severity: 4
6.4.3.3 b) 3)	This activity is an integral part of 6.4.3.3 a) Severity: 4
6.4.3.3 b) 4)	This paragraph has alternative “design solutions” – whatever that means – being evaluated, without having been created. Evaluation of, and selection between, alternative designs (or alternative solution descriptions, the same thing except for non-developmental solutions), each capable of meeting requirements, is a very good idea provided the expected benefit exceeds the expected cost of the evaluation. The decision-making between design alternatives in is Project Processes – not helpful. Severity: 6
6.4.3.3 c)	The words are somewhat obtuse and detract from the usefulness of the standard. Severity: 2
6.4.4.1 Purpose (of Implementation Process)	6.4.4.1 Purpose (of Implementation Process) Specified behavior and interfaces are implementation constraints (all requirements are implementation constraints). Severity: 1

6.4.4.1 Purpose (of Implementation Process)	The term "design requirement" is used for the first time, apparently to mean "requirement". If the term "design requirement" is used to mean "requirement", every requirement is a "design requirement". That is not really very useful! Severity: 4
6.4.4.1 Purpose (of Implementation Process)	This paragraph states: "This process results in a system element that satisfies specified design requirements through verification and stakeholder requirements through validation." The statement is absurd – verification provides evidence, not the means, of satisfaction of requirements. Validation provides evidence, not the means, of satisfaction of stakeholder needs. Severity: 4
6.4.4.2 a) Outcomes (of the Implementation Process)	The definition of an implementation strategy (i.e. implementation system conceptual design) is stated to be an outcome of this process. But 5.1.4 correctly states that an enabling system can be considered to be a "system-of-interest" to an entity responsible for its implementation, whilst 5.1.3 correctly emphasizes the recursive nature of development of systems and subsystems. An implementation system, e.g. production system, is a system like any other system, but is also a subsystem of the bigger system, of which production is a part of the solution. The process of defining an implementation system design is covered therefore under 6.4.3. Oh, what a mess in this paragraph! In applying this muddled part of the standard, Implementation Process should be confined to 6.4.4.2 c) and 6.4.4.2 d) Severity: 7
6.4.4.3 a)	Please see comments on 6.4.4.2 a). The valid content of 6.4.1 to 6.4.3 inclusive relates. Severity: 7
6.4.4.3 b) 2)	A system element does not normally meet supplier agreements, etc., unless design is specified in requirements. The aim is that the system element be consistent with meeting such agreements. Severity: 3
6.4.5.1 Purpose (of Integration Process)	The statement needs to make it clear that the purpose is to assemble the system in development, and excludes assembly in production, for cases where more than one instance of a system is to be produced. Severity: 5
6.4.5.2 Outcomes (of Integration Process)	The definition of an integration strategy (i.e. integration system conceptual design) is stated to be an outcome of this process. But 5.1.4 correctly states that an enabling system can be considered to be a "system-of-interest" to an entity responsible for its implementation, whilst 5.1.3 correctly emphasises the recursive nature of development of systems and subsystems. An integration system is a system like any other system, but is also a subsystem of the bigger system, of which integration is a part of the solution. The process of defining an integration system design is therefore covered under 6.4.3. Oh, what a mess in this paragraph! In applying this muddled part of the standard, Integration Process should be confined to 6.4.5.2 c) and 6.4.5.2 d) Severity: 7
6.4.5.3 a)	Please see comments on 6.4.5.2 a). The valid content of 6.4.1 to 6.4.3 inclusive relates. Severity: 7

6.4.6.1 Purpose (of Verification Process)	The term “design requirement” is used, apparently to mean “requirement”. If the term “design requirement” is used to mean “requirement”, every requirement is a “design requirement”. That is really unhelpful! Severity: 4
6.4.6.1 Purpose (of Verification Process)	Limiting the Verification Process to system verification is illogical, and inconsistent with good practice in engineering. All work products are candidates for verification. Work products should be verified where the risk reduction benefit exceeds the verification cost, and limited resources cannot be employed in a more beneficial way. Severity: 9
6.4.6.2 Outcomes (of Verification Process)	The definition of a verification strategy (i.e. verification system conceptual design) is stated to be an outcome of this process. But 5.1.4 correctly states that an enabling system can be considered to be a “system-of-interest” to an entity responsible for its implementation, whilst 5.1.3 correctly emphasises the recursive nature of development of systems and subsystems. A verification system is a system like any other system, but is also a subsystem of the bigger system, of which verification is a part of the solution. The process of defining a verification system design is covered therefore under 6.4.3. Oh, what a mess in this paragraph! In applying this muddled part of the standard, Verification Process should be confined to 6.4.6.2 c) and 6.4.6.2 d) Severity: 7
6.4.6.3 Activities and Tasks (of the Verification Process)	6.4.6.3 Activities and Tasks (of the Verification Process) It is unfortunate that the standard makes no provision for verification requirements, meaning that verification design is carried out in a vacuum. The practical effect is usually either insufficient verification, or excessive verification. Either way, stakeholder value is reduced. Severity: 8
6.4.6.3 a)	Please see comments on 6.4.6.2 a). The valid content of 6.4.1 to 6.4.3 inclusive relates. Severity: 7
6.4.6.3 b) 2)	Again the term “design requirements” is used. Previous comments apply.
6.4.7.1 Purpose (of Transition Process)	By including installation of relevant enabling systems in the purpose of the Transition Process, ISO/IEC 15288:2008 double-counts the enabling systems with respect to transition. This is because 5.1.4 (correctly) states that an enabling system can be considered to be a “system-of-interest”, and each enabling system, therefore, is also subject to the transition process in its own right. What a mess! Severity: 7
6.4.7.2 Outcomes (of the Transition Process)	The definition of a transition strategy (i.e. transition system conceptual design) is stated to be an outcome of this process. But 5.1.4 correctly states that an enabling system can be considered to be a “system-of-interest” to an entity responsible for its implementation, whilst 5.1.3 correctly emphasises the recursive nature of development of systems and subsystems. A transition system is a system like any other system, but is also a subsystem of the bigger system, of which transition is a part of the solution. The process of defining a transition system design is covered therefore under 6.4.3. Oh, what a mess in this paragraph! In applying this muddled part of the standard, Transition Process should be confined to 6.4.7.2 b) to f). Severity: 7
6.4.7.3 a)	Please see comments on 6.4.7.2. The valid content of 6.4.1 to 6.4.3 inclusive relates. Severity: 7

6.4.7.3 b) 6)	This paragraph doesn't allow for enabling systems that have ceased to be relevant, e.g. development systems and production systems. Worse, again this paragraph is at odds with 5.1.4 which correctly states that an enabling system can be considered to be a "system-of-interest". Showing that an end-use system is sustainable by the relevant enabling systems is an act of system integration of the parent (capability) system, not an act of transition of the system-of-interest. Severity: 7
6.4.8.1 Purpose (of Validation Process)	Limiting the Validation Process to system validation is illogical, and inconsistent with good practice in engineering. All work products are candidates for validation. Work products should be validated where the risk reduction benefit exceeds the validation cost, and limited resources cannot be employed in a more beneficial way. Severity: 7
6.4.8.2 Outcomes (of Validation Process)	The definition of a validation strategy (i.e. validation system conceptual design) is stated to be an outcome of this process. But 5.1.4 correctly states that an enabling system can be considered to be a "system-of-interest" to an entity responsible for its implementation, whilst 5.1.3 correctly emphasises the recursive nature of development of systems and subsystems. A validation system is a system like any other system, but is also a subsystem of the bigger system, of which validation is a part of the solution. The process of defining a validation system design is covered therefore under 6.4.3. Oh, what a mess in this paragraph! In applying this muddled part of the standard, the Validation Process should be confined to 6.4.8.2 b) to d) Severity: 7
6.4.8.3 a)	Please see comments on 6.4.8.2. The valid content of 6.4.1 to 6.4.3 inclusive relates. Severity: 7
6.4.8.3 b) 2)	The reference for validation should be need, not requirements. Otherwise, validation ceases to serve a purpose if verification is performed. In making this comment, I am using all terms in accordance with the Oxford English Dictionary, and also reflecting very widespread practice in engineering, including areas subject to regulation such as aviation and medical products.
6.4.8.3 b) 4)	Diagnosing the cause of invalidity is a problem-solving action outside of the scope of validation. Diagnosing the cause of invalidity is within the scope of 6.4.1 and 6.4.2 if caused by defective requirements, 6.4.3 if caused by defective design, and other processes if caused by failures in those processes.
6.4.9.1 Purpose (of Operation Process)	The reference to analysis of operational problems is not (or should not be) within the scope of the Operation Process. Diagnosing the cause of operational problems is within the scope of 6.4.1 and 6.4.2 if caused by defective requirements, 6.4.3 if caused by defective design, and other processes if caused by failures in those processes.
6.4.9.2 Outcomes (of Operation Process)	The reference to analysis of operational problems is not (or should not be) within the scope of the Operation Process. Diagnosing the cause of operational problems is within the scope of 6.4.1 and 6.4.2 if caused by defective requirements, 6.4.3 if caused by defective design, and other processes if caused by failures in those processes. 6.4.9.2 Outcomes (of Operation Process) The definition of strategy for operation is stated to be an outcome of this process. However, designing how to operate a system and definition of skills and other attributes required of operators should be done integral with the design of the technology aspects of the solution. Also, it is commonplace to place operating procedures within the boundary of the system, but operators outside of that boundary. For example, for an aircraft system developed and supplied by Airbus or Boeing, the operating procedures are inside the boundary but the operators (aircrew) are outside of the boundary. By contrast, the aircrew are system elements within an air transportation system. In applying this muddled part of the standard, the Operation Process should be confined to 6.4.9.2 b) and d) Severity: 10
6.4.9.3 a)	Please see comments on 6.4.9.2. with respect to operational infrastructure. The valid content of 6.4.1 to 6.4.3 inclusive relates. Please see also the entry relating to a missing System Management Process. Severity: 9

6.4.9.3 c)	Please see comments on 6.4.9.2. with respect to operational infrastructure. The valid content of 6.4.1 to 6.4.3 inclusive relates. Severity: 9
6.4.10.2 Outcomes (of Maintenance Process)	The definition of a maintenance strategy (i.e. maintenance system conceptual design) is stated to be an outcome of this process. But 5.1.4 correctly states that an enabling system can be considered to be a “system-of-interest” to an entity responsible for its implementation, whilst 5.1.3 correctly emphasises the recursive nature of development of systems and subsystems. A maintenance system is a system like any other system, but is also a subsystem of the bigger system, of which maintenance is a part of the solution. The act of designing the maintenance system is covered therefore under 6.4.3. Oh, what a mess in this paragraph! In applying this muddled part of the standard, the Maintenance Process should be confined to 6.4.10.2 c) to f) inclusive. Severity: 10
6.4.10.3 a)	Please see comments on 6.4.10.2. with respect to the maintenance system. The valid content of 6.4.1 to 6.4.3 inclusive relates. Severity: 10
6.4.10.3 b) 1)	Obtaining the enabling systems, system elements and services to be used during maintenance is not an act of maintenance, but an act of (maintenance) system integration. Severity: 4
6.4.10.3 b) 2)	Implementing problem reporting and incident recording is not an act of maintenance, but an act of (maintenance) system integration. Severity: 4
6.4.10.3 b) 5) NOTE	The acquisition, training and accreditation of personnel to maintain operator numbers and skills is not an act of maintenance, but an act of system re-implementation. Severity: 3
6.4.11.1 Purpose (of Disposal Process)	If the Disposal Process is limited in scope to ending the existence of a system entity, the set of life cycle processes defined in ISO/IEC 15288:2008 does not allow for sale or other transfer of responsibility for a system. That omission could be covered within a (missing) System Management process.
6.4.11.2 Outcomes (of Disposal Process)	The definition of a disposal strategy (i.e. disposal system conceptual design) is stated to be an outcome of this process. But 5.1.4 correctly states that an enabling system can be considered to be a “system-of-interest” to an entity responsible for its implementation, whilst 5.1.3 correctly emphasises the recursive nature of development of systems and subsystems. A disposal system is a system like any other system, but is also a subsystem of the bigger system, of which the means of disposal is a part of the solution. The act of designing the disposal system is covered therefore under 6.4.3. Oh, what a mess in this paragraph! In applying this muddled part of the standard, the Disposal Process should be confined to 6.4.10.2 c) to e) inclusive. Severity: 10
6.4.11.3 a)	Please see comments on 6.4.11.2. with respect to the disposal system. The valid content of 6.4.1 to 6.4.3 inclusive relates. Severity: 10

6.4.11.3 b) 1)	Acquiring the enabling systems or services to be used during disposal is not an act of disposal, but an act of (disposal) system implementation or integration. The paragraph conspicuously trivialises the implementation of infrastructure for use in disposing of the elements of the system-of-interest. Severity: 6
General Comment	The standard lacks a system management process. System management is the discipline and activity concerned with managing the operation, sustainment, evolution, and retirement of a system.

Definitions to Close on

Model, MBD, MBE, MBIT, MBSE, MBT, MDA, MDD, MDE

Model: Graphical, mathematical (symbolic), physical, or verbal representation or simplified version of a concept, phenomenon, relationship, structure, system, or an aspect of the real world.
Source: http://www.businessdictionary.com/definition/model.html

Model: A representation of a system that allows for investigation of the properties of the system and, in some cases, prediction of future outcomes.
Source: http://www.investorwords.com/5662/model.html

Model: A physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process.
Source: DoD5000.59-M 1998

Model Based Definition (MBD): A 3D annotated model and its associated data elements that fully define the product definition in a manner that can be used effectively by all downstream customers in place of a traditional drawing.
Source: DEDMWG-MBE (2010-Aug), DoD Engineering Drawing and Modeling Working Group (DEDMWG)

Model Based Design (MBD): A development approach which declares the 3D CAD model as the record of authority and is the source for which all other documentation flows. This design method facilitates MBE, by emphasizing digital CAD file use for collaboration at the beginning of development.
Source: NAS 9300-007 (2008) LOTAR, DEDMWG-MBE (2010-Aug), DoD Engineering Drawing and Modeling Working Group (DEDMWG)

Model-based design (MBD): Model-Based Design (MBD) is a mathematical and visual method of addressing problems associated with designing complex control, signal processing and communication systems. It is used in many motion control, industrial equipment, aerospace, and automotive applications. Model-based design is a methodology applied in designing embedded software.
Source: http://en.wikipedia.org/wiki/Model-based_design

Model Based Development (MBD): A technique of simulating system performance off-line, and then generating code from the simulation.
Source: http://en.wikipedia.org/wiki/VisSim#Use_of_model-based_development

Model Based Enterprise (MBE): Fully integrated and collaborative environment founded on 3D product definition detailed and shared across the enterprise to enable rapid, seamless, and affordable deployment of products from concept to disposal.
Source: NAS 9300-007 (2008) LOTAR, DEDMWG-MBE (2010-Aug), DoD Engineering Drawing and Modeling Working Group (DEDMWG)

Model-based engineering (MBE): MBE is an approach to engineering in which models:

- are an integral part of the technical baseline;
- evolve throughout the acquisition life cycle;
- are integrated across all program disciplines (e.g., systems engineering, operations analysis, software engineering, hardware engineering, manufacturing, logistics, etc.); and
- can be shared and/or reused across acquisition programs, including between Government and Industry stakeholders.

Source: NDIA, Final Report of the Model Based Engineering (MBE) Subcommittee, 10 February 2011

Model Based Integration and Testing (MBIT): A method of system integration and testing which allows for integration of models of not yet realized components (e.g. mechanics, electronics, software) with available realizations of other component.
Source: “Model-Based Support for Integration and Testing of a Multi-Disciplinary Industrial System, www.esi.nl/... /20060920_EuSEC2006_v11.pdf

Model-based systems engineering (MBSE): Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.
Source: INCOSE SE Vision 2020 (INCOSE-TP-2004-004-02), Sept 2007

Model-based systems engineering (MBSE): Model-based systems engineering (MBSE) is an umbrella term referring to the formalized application of modeling (usually logical modeling) to support one or more of system requirements, design, analysis, verification and validation activities, desirably beginning in the conceptual design phase and continuing throughout development and later life cycle phases.
Source: Robert Halligan, after INCOSE SE Vision 2020 (INCOSE-TP-2004-004-02), Sept 2007

Model-based testing (MBT): Model-based testing involves checking whether a system or component realization behaves in accordance with its model. This means that the model is taken as the component specification to which the realization must conform. Checking this conformance is done by means of testing.
Source: “Tangram: Model-based integration and testing of complex high-tech systems”, Embedded Systems Institute, Eindhoven, the Netherlands, 2007, ISBN: 978-90-78679-02-8

Model Driven Architecture® (MDA®): MDA is a way to organize and manage enterprise architectures, supported by automated tools and services for both defining the models and facilitating transformations between different model types. MDA was launched by the Object Management Group (OMG) in 2001. Both the term and the acronym are OMG-registered IP.

The MDA approach, rooted in software engineering, defines system functionality using a platform-independent model (PIM) using an appropriate domain-specific language (DSL). Then, given a platform model corresponding to CORBA, .NET, the Web, etc., the PIM is translated to one or more platform-specific models (PSMs) that computers can run. This requires mappings and transformations and should be modeled too. The PSM may use different DSLs, or a General Purpose Language (GPL) like Java, C#, PHP, Python, etc. Automated tools generally perform this translation.

Note that the reference to “architecture” in MDA is the architecture of the models, not the architecture of the system being developed.
Source: Robert Halligan, aided by various sources.
Additional information

Model-Driven Development (MDD): MDD (Model-Driven Development) approaches involve the automatic generation of software products by means of the transformation of the defined models into the final program code.
Source: http://ec.europa.eu/information_society/events/cf/ss0911/item-display.cfm?id=7237

Model-Driven Engineering (MDE): Model Driven Engineering is an approach to software engineering aims to raise the level of abstraction in program specification and increase automation in program development by using models at different levels of abstraction for developing systems, thereby raising the level of abstraction in program specification. An increase of automation in program development is achieved by using executable model transformations. Higher-level models are transformed into lower level models until the model can be made executable, using either code generation or model interpretation.
Source: Robert Halligan, after Johan den Haan at http://www.theenterprisearchitect.eu/archive/2009/01/15/mde---model-driven-engineering----reference-guide

Model-Driven Engineering (MDE): Model-driven engineering (MDE) is an umbrella term referring to a system development approach which focuses on creating and exploiting domain models (that is, abstract representations of the knowledge and activities that govern a particular application domain, and excluding solution technology).
Source: after http://en.wikipedia.org/wiki/Model-driven_engineering

Model-Based or Model-Driven? – A Note from Robert
As can be seen from the definitions, a purposeful distinction can be made between model-based and model-driven, although, in practice, the terms are sometimes used interchangeably. The distinction is:
Model-based = The model is an important aid but is not the artifact from which downstream development is primarily conducted;
Model-driven = The model is the artifact from which downstream development is primarily conducted.
This is a useful distinction.

PPI News (see www.ppi-int.com)

PPI at IS2012

PPI recently sponsored and exhibited at the INCOSE International Symposium IS2012, in one of the worlds most historic and breathtaking cities - Rome, Italy. Delegates attended from around the world, including Australia, North America, South America and Africa. IS2012 attracted many of the world's leading systems engineers, providing opportunities to network and at the same time learn of the latest in thinking, tools, methodologies and concepts. The conference involved, each day, 9 simultaneous tracks, a renowned plenary speaker, working group activities, turotials and business meetings. IS2012 provided overall about 116 papers, 14 panels, 18 tutorials and more than 40 poster presentations. A detailed report will follow in the next edition of SyEN.

Cognitive Systems Tutorial

Dr. Gavan Lintern has developed a new tutorial on cognitive work analysis. For those who do not know about cognitive work analysis, it is a multistage framework for analyzing cognitive functions and processes in a distributed, socio-technical system. This framework is unique among cognitive systems engineering methods in that it takes a systems view in approaching the analysis of cognition. For those who have attended PPI's cognitive systems engineering course, this tutorial offers a concise summary of the material covered in the second and third days on cognitive work analysis and adds some new concepts to the explanations of the later stages of the analysis. In particular, there is a fairly extensive section on social organization analysis.

The tutorial does not require any prior knowledge of cognitive work analyis.

The tutorial can be downloaded without charge from www.cognitivesystemsdesign.net. Go to the workshops page before login or to the tutorials page or the workshops page after login.

PPI Events (see www.ppi-int.com)

Systems Engineering Public 5-Day Courses

Upcoming Locations Include:

- Adelaide, Australia
- Brisbane, Australia
- Melbourne, Australia
- Rio de Janeiro, Brazil
- Munich, Germany

Requirements Analysis and Specification Writing Public Courses

Upcoming Locations Include:

- Amsterdam, The Netherlands
- Adelaide, Australia
- Melbourne, Australia
- Las Vegas, USA

Software Engineering Public 5-Day Courses

Upcoming Locations Include:

- Sydney, Australia
- Pretoria, South Africa
- Amsterdam, The Netherlands

OCD/CONOPS Public Courses

Upcoming Locations Include:

- Brasilia, Brazil
- Pretoria, South Africa
- Las Vegas, USA

Cognitive Systems Engineering Courses

Upcoming Locations Include:

- Adelaide, Australia
- Las Vegas, USA

CSEP Preparation Course (Presented by PPI subsidiary Certification Training International)

Upcoming Locations Include:

- Las Vegas, USA
- Austin, USA
- Munich, Germany

PPI Upcoming Participation in Professional Conferences

PPI will be participating in the following upcoming events. We look forward to chatting with you there.

- 9th Annual INCOSE SA Conference | Exhibiting | South Africa (27 - 29 August, 2012)
- 2nd European Defence Conference | Participating | Prague, Czech Republic (9 - 10 October, 2012)
- New Zealand Defence Industry Association Forum 2012 | Exhibiting | Wellington, New Zealand (16 - 17 October, 2012)
- Land Warfare Conference 2012 | Exhibiting | Melbourne, Australia (22 - 26 October, 2012)
- MilCIS 2012 Conference | Participating | Canberra, Australia (6 - 8 November, 2012)
- CSD&M 2012 Conference | Participating | Paris, France (12 - 14 December, 2012)

Kind regards from the SyEN team:
Robert Halligan, Managing Editor, email: rhalligan@ppi-int.com
Ralph Young, Editor, email: ryoung@ppi-int.com
Stephanie Halligan, Production, email: shalligan@ppi-int.com

Project Performance International
2 Parkgate Drive, Ringwood, Vic 3134 Australia
Tel: +61 3 9876 7345
Fax: +61 3 9876 2664
Tel Brasil: +55 11 3230 8256
Tel UK: +44 20 3286 1995
Tel USA: +1 888 772 5174
Web: www.ppi-int.com
Email: contact@ppi-int.com

Tell us what you think of SyEN: email to contact@ppi-int.com