26th Annual INCOSE International Symposium (IS 2016)
Edinburgh, Scotland, UK, July 18-21, 2016

Towards a Quantitative Framework for Evaluating the Expressive Power of Conceptual System Models

Yaniv Mordecai
Faculty of Industrial Engineering & Management,
Technion – Israel Institute of Technology,
Haifa, Israel
YanivMor@technion.ac.il

Dov Dori
Faculty of Industrial Engineering & Management
Technion – Israel Institute of Technology,
Haifa, Israel
Dori@ie.technion.ac.il

Abstract. Conceptual models describe, explain, and specify the function, concept, structure, and behavior of complex systems. Quantifying the contribution of conceptual models to stakeholder understanding of the systems-of-interest has been a great challenge. This difficulty has hindered justifying the use of formal modeling and simulation of complex systems and the adoption of a holistic model-based systems engineering (MBSE) approach. The informativity of a model is the value of the information it conveys. Conceptual system model informativity is a key performance indicator for MBSE. We introduce MIA – Model Informativity Analysis – a quantitative, utility-based, prescriptive approach for boosting conceptual models' expressive power and measuring the value of the information they provide. We define an integrated informativity index, "I^3", which aggregates model scores of diverse informativity-enhancing factors. We demonstrate various aspects of MIA with Object Process Methodology, OPM – a model-based systems engineering paradigm and ISO-19450 standard. OPM caters to quantitative informativity analysis due its formality, comprehensive function-structure-behavior modeling, and bimodal equivalent graphical-textual representation.

Keywords. Conceptual Modeling, Informativity, Object-Process Methodology, Utility Theory, Information Theory.

Acronym Glossary

Acronym	Full Term	Definition
Ad-hoc terms		
IEF	informativity-enhancing factor	attribute of messages whose value affects the amount of information that can be extracted from a message
INF	informativity figure	number in [0..1] or [-1..1] indicating the information addition by a model fact due to a specific informativity-enhancing factor
I^3	Integrated Informativity Index	aggregate measure of the total amount of information carried by a model
MIA	Model Informativity Analysis	quantitative, utility-based framework for measuring and analyzing the informativity and information-based utility of formal conceptual models
Reference Terms		
MBSE	Model-based Systems Engineering	
OPM	Object Process Methodology	
SysML	Systems Modeling Language	
UML	Unified Modeling Language	

Introduction

Conceptual models are valuable for describing systems, phenomena, and problems, and the key role such models plays in science and engineering has been recognized long ago (Rosenblueth & Wiener, 1945). Such models, be they whiteboard or notebook sketches, textbook or document drawings, or digitally-managed formal data structures, are used by their originators for several reasons (Cherubini, Venolia, Deline, & Ko, 2007; D. Embley & Thalheim, 2011): (i) Making sense: clarifying, explaining a concept, rationalizing or grounding it, or understanding "how things look and work"; (ii) Collaboration: sharing and communicating ideas with colleagues, brainstorming to solve conceptually-challenging problems, and experimenting with new ideas; (iii) Analysis: organizing and architecting, applying or searching for patterns, evaluating or optimizing the design, hypothesizing and performing qualitative simulations for answering "what-if" questions; (iv) Implementation or construction: evolving models into working prototypes, features, or whole systems, providing blueprints for detailed design, manufacturing, assembly, or installation; (v) Testing, demonstration and simulation: validating and verifying sequences of events, checking consistency, experimenting with potential scenarios and results, illustrating how a system is intended to work and what problems it might encounter; (vi) Recording and documenting: capturing the current, intended, or expected state of affairs; (vii) Monitoring and Control: tracking, evaluating, and regulating the behavior of systems with respect to their models, assuring compliance with expected or intended behavior.

The formality of models is critical for their encoding, verification, validation, consistency checking, reproduction, and comparison with other models. Formal conceptual modeling is based on four principles: (i) Mapping: the model is a representation of an origin: a system, a product, a service, or a phenomenon; (ii) Purpose: the model serves one or more purposes, roles, or goals; (iii) Language: the model uses formal language to represent the origin and fulfill its purpose; (iv) Value: the model provides utility, benefit, or positive value (benefit at cost) to its authors and readers.

The value of a model – the utility or benefit at cost that it provides – is significantly affected by the amount and quality of information that the model expresses, holds, or stores. Therefore, model informativity – a measure of the value of information that a model conveys – is potentially a highly useful index for the evaluation and comparison of models and model versions.

The information-theoretic approach defines the amount of information carried in a message over a communication channel as the entropy of the data with respect to its distribution (Howard, 1966, 1968). The formulation of information functions has long intrigued decision scientists, due to the well-grounded idea that information assists decision makers by increasing their expected utility. The consideration of informativity as utility has been discussed by Azrieli & Lehrer (2008).

Although the benefit of measuring model informativity has been recognized, research on conceptual model evaluation has been surprisingly scarce. A definition of the value, utility, or informativity of conceptual models has remained vague and unclear, and therefore the value itself has been non-quantifiable and immeasurable. Systems engineers, architects, and designers rarely ask questions such as "How informative is a given model?", "How much information does (or can) a model convey?", or "Which of two models of the same problem is more informative?" The common assumption is that such questions are likely too difficult to answer. Assessing models for their capability to faithfully specify the systems they are supposed to represent has remained largely intuitive, if it is at all pursued.

The purpose of this paper is to introduce a framework for evaluating the informative utility of conceptual system models. Our Model Informativity Analysis approach, MIA, relies on the premise that models have value primarily due to the fact that they capture, organize, and provide information and

represent knowledge. This approach promoted expected information analysis rather than expected utility analysis, as the utility of information was assumed to be difficult to quantify (Bernardo, 1979). In the following section we review literature on model evaluation. We then elaborate on the taxonomy and analytics of MIA. Finally, we discuss the validity and applicability of the MIA approach.

Literature Review

Models vary in quality and fidelity. Partially-understood systems suffer from some discrepancy between the models that describe them and their actual behavior and structure. Modeling of systems under insufficient and imprecise a-priori information conditions is a tradeoff between approximation and complexity (Maciejowski, 1979), as well as between clarity and completeness (Dori, 2002). This discrepancy is the basis for the potential multiplicity of and variability among several models of the same problem (Goldstein, Tech, Va, & Rougier, 2008). Akaike's Information Criterion (AIC) is an estimate of a statistical predictive model's efficiency, $AIC = -2 \cdot \log L(\hat{\theta}) + 2k$ where $L(\hat{\theta})$ is the likelihood function and k is the number of free parameters (Akaike, 1974). AIC is useful for comparing statistical models for the same stochastic phenomenon. Variations and extensions of AIC for evaluating the information complexity of statistical models are available in the literature (Bozdogan, 2000).

The study of quality factors of complex conceptual system models and discriminators of models of the same problem is scarce. Studies of the informativity of knowledge representations – models, reports, and formalisms – concern cognitive comparisons (Bowdle & Gentner, 1997), stock analyst reports (Frankel, Kothari, & Weber, 2006) and formal logic (Trentelman, 2009). Most studies on model quality and informativity are tailored to specific reasoning models, whose applicability to general systems science and engineering is limited.

A model is an information system and a knowledge base in the sense that it captures and provides information and knowledge—the ability to use information—about a domain or a system-of-interest. The model's informativity can be assessed using structural and functional measures. Structural measures are based on the structure of the model and the information or knowledge it represents. Functional measures of knowledge refer to the model's usefulness for task performance and problem-solving. Structural and functional measures can be qualitative or quantitative, problem-specific or task-specific (Reich, 1995). These notions have been implemented in machine learning (Reich & Barai, 1999) and general design theory (Reich, 2002).

Several kinds of models carry distinct informative value to system or problem stakeholders. These include (1) graphical or analytical general-purpose parameter relationship models, (2) experimentally- or analytically-calibrated models of well-defined processes, and (3) scientifically-grounded models, in which process parameters are associated with laws of physics (Gnoevoi & Chesnokov, 2003). Models of the first kind are considered slightly informative due to the imprecise cause-effect relations they express. Models of the second kind are more informative, since they can help answer quantitative questions on the behavior of the system. The third kind of models are the most informative, since they characterize and quantify system behavior and structure, clarifying fundamental cause-effect relations.

Specification patterns enhance statement formality and uniformity. A taxonomy of language-independent specification patterns was created in the SAnToS Laboratory (Dwyer, Avrunin, & Corbett, 1998). Specification patterns were also proposed for real-time systems (Konrad & Cheng, 2005) and probabilistic systems (Grunske, 2008). In principle, a model in any formal conceptual modeling language is amenable to being represented as a set of model facts from which formal model statements can be derived. Admittedly though, this is not easy for modeling languages such as UML or

SysML, and may result in dozens of formal patterns for each one of the 14 diagram kinds (Funes & George, 2003).

Model Informativity Analysis – MIA

Acknowledging the importance of models as a source of valuable information and knowledge, we argue that model informativity measurement and analysis are critical for model utility evaluation, model and model version comparison, and opportunities to identify model informativity enhancements. In this section, we introduce Model Informativity Analysis, MIA – a quantitative, utility-based framework for measuring and analyzing the informativity and information-based utility of formal conceptual models. MIA provides an informativity index based on attributes of model statements (or model facts) that enhance the informative value of the model. Adding domain-specific criteria, as well as filtration of irrelevant ones, are allowed. We envision a quantitative MIA-based evaluation mechanism built into software tools for conceptual system modeling in various languages, which provides for monitoring changes in model informativity throughout the system lifecycle from inception all the way to retirement.

The idea behind MIA is independent of the conceptual modeling language underlying the model it evaluates. In this paper we focus on using MIA to evaluate models in Object Process Methodology, OPM (Dori, 2002, 2016). OPM is a holistic conceptual modeling language and a model-based systems engineering paradigm, which has been adopted as ISO-19450 (ISO/TC 184, 2015). OPM enables holistic modeling of function, structure, and behavior of complex systems, and provides a formal textual specification in Object-Process Language (OPL), a context-free grammar-based subset of natural English. An OPL sentence (or sentence part) expresses textually each construct expressed graphically in one of the hierarchically-organized Object-Process Diagrams (OPDs). OPM provides a formal and potentially-comprehensive specification that allows for informativity and utility analysis and evaluation of models as they evolve, as well as for comparing two or more models specifying the same system.

MIA aims to produce the aggregate, representative Integrated Informativity Index of a conceptual model, $I^3(M)$, based on a variety of informativity-enhancing factors (IEFs). The IEFs are clustered into four groups: (i) specification pattern, (ii) specification uncertainty, (iii) meta-specification, and (iv) specification management. We elaborate on each group in the following subsections.

Specification Patterns

Formal modeling mandates the use of a predefined formal language having three agreed-upon foundations: (i) syntax – a set of elements: entity building blocks, relations among them, their symbols, and legal ways in which they can be put together, (ii) semantics – the deep meaning of syntactically well-organized elements, and (iii) ontology – a conceptual organization of the domain and its building blocks for which the modeling formalism applies (Dori, 2011). Any model consists of a set of (graphical and/or textual) statements that collectively and synergistically form the specification of the system, product, service, phenomenon, or problem that the model aims to specify.

OPM's formal statements are called model facts, but they also describe non-factual statements, such as assertions or requirements that are expected to become facts. Model facts are expressed graphically as OPD constructs and textually as semantically equivalent OPL sentences or sentence parts. Each OPL sentence kind is based on a single specification pattern, which is one of 21 patterns. These patterns, listed in Table 1, are classified into four groups: a) thing definition, b) structural link, c) procedural link, and d) precedence link. Model fact patterns are modular and extensible with details and refinements.

Table 1. OPM model fact specification patterns

Specification Group	Specification Pattern	Basic Informativity	Refinements and Extensions
Thing Definition	1. Object Definition	i_{01}	Affiliation detailing (r_{01}) Essence detailing (r_{02}) Type specification (r_{03})
	2. Process Definition	i_{02}	Affiliation detailing (r_{01}) Essence detailing (r_{02})
	3. State Set Definition	i_{03}	Semantic "inexistent" state (r_{0401}) Semantic "disrupted" state (r_{0402}) Semantic "unknown" state (r_{0403})
	4. State Description	i_{04}	Initial state indicating (r_{0501}) Final/Default indicating (r_{0502})
Structural Link	5. Aggregation-Participation	i_{05}	Cardinality indicating (r_{06}) Parent state specifying (r_{07}) Child state specifying (r_{08})
	6. Exhibition-Characterization	i_{06}	Multiplicity (r_{09}) Cardinality indicating (r_{06}) Parent state specifying (r_{07}) Child state specifying (r_{08})
	7. Generalization-Specialization	i_{07}	Parent state specifying (r_{07}) Child state specifying (r_{08})
	8. Classification-Instantiation	i_{08}	Parent state specifying (r_{07}) Child state specifying (r_{08})
	9. Unidirectional Tagged Relation	i_{09}	Relation Specifying (r_{10}) Cardinality indicating (r_{06}) Thing1 state specifying (r_{07}) Thing2 state specifying (r_{08})
	10. Bidirectional Tagged Relation	i_{10}	Relation Specifying (r_{10}) Cardinality indicating (r_{06}) Pointing state specifying (r_{07}) Pointed state specifying (r_{08})
Procedural Link	11. Agent Link	i_{11}	Path specifying (r_{11})
	12. Resource Link	i_{12}	Path specifying (r_{12}) Multiplicity (r_{09}) Logic (AND/OR/XOR) (r_{12}) State Specifying (r_{07})
	13. Result Link	i_{13}	Path specifying (r_{11}) Multiplicity (r_{09}) Logic (AND/OR/XOR) (r_{12}) State Specifying (r_{07})
	14. Consumption Link	i_{14}	Path specifying (r_{11}) Multiplicity (r_{09}) Logic (AND/OR/XOR) (r_{12}) State Specifying (r_{07})
	15. Effect Link	i_{15}	Path specifying (r_{11}) Multiplicity (r_{09}) Logic (AND/OR/XOR) (r_{12})
	16. Transformation	i_{16}	Path specifying (r_{11}) Multiplicity (r_{09}) Logic (AND/OR/XOR) (r_{12})
	17. Instrument Event	i_{17}	State Specifying (r_{07}) State Change Dependency (r_{13})
	18. Condition Link	i_{18}	Path specifying (r_{11}) Multiplicity (r_{09}) Logic (AND/OR/XOR) (r_{12}) State Specifying (r_{07})
Precedence Link	19. Invocation Link	i_{19}	Path specifying (r_{11})
	20. Exception Link	i_{20}	Path specifying (r_{12})
	21. In-zooming	i_{21}	N/A

Each specification pattern is extensible with up to four of the 13 syntactic or semantic extensions listed in Table 1 as r_{01} through r_{13}. Thing definitions, for instance, can include affiliation (to distinguish systemic things from environmental ones) and essence (to distinguish informatical things from physical ones). Procedural links to or from objects can be refined to originate or terminate at specific states of

that object. Procedural links can be limited to a specific path – a thread label that forces the process to exit on procedural links that have the same path label as the one with which the process was entered (Dori, 2016). Procedural patterns combine multiple links of the same pattern under logical conditions.

Each specification pattern has a certain level of importance, based on such considerations as its criticality for model simulation, order of execution, or effect on final results. Some patterns are mostly declarative and meant for clarification. They are therefore less significant than patterns that control the behavior of the model. For example, generalization-specialization (gen-spec) relations are less significant than invocation links. This is because gen-spec relations serve mostly for informing purposes and design-time inheritance of attributes and operations from ancestor, general things. Invocation links determine process execution order and dependency during model simulation and therefore affect the behavior of the model. On top of the basic importance level, each model-fact can provide additional information by utilizing its possible extensions. Each specification pattern has a range of possible values of its reflective informativity figure (INF), based on the basic informativity rate $i_k, k \in \{1,..,21\}$, and the implemented extensions and refinements $r_m, m \in \{1,..,13\}$. INFs are normalized in [0..1] for each specification pattern and each valid combination of extension utilizations, such that the amount of information provided by each model fact is finite and relative.

The basic informativity and refinement informativity values of each pattern and extension are different relative to the others. While it is more straightforward to state that the basic informativity of procedural links is higher than thing definitions ($\min\{i_{12}, i_{13}, i_{14}, i_{15}, i_{16}, i_{17}, i_{18}\} \geq \max\{i_{01}, i_{02}, i_{03}, i_{04}\}$), it is more difficult to determine whether, for example, a condition link (i_{18}) is more important than an event link (i_{15}). Relative importance and derived basic and refined informativity of specification patterns and extensions are therefore beyond the scope of this paper, and are part of ongoing research.

Specification Uncertainty

Our perception of the value of information combines an information-theoretic approach with a utility-theoretic approach. Each message carries information as part of a stream of messages – a communication session. Similarly, a model is a set of communicated messages—model facts. The deterministic nature of conceptual model-based specification leaves little room for stochastic and uncertainty-related specification.

There are two essential types of uncertainty: *aleatory* and *epistemic*. Aleatory uncertainty originates from natural variability and randomness of some phenomenon or event, while epistemic uncertainty originates from lack of sufficient knowledge or information about the system at work (Bedford & Cooke, 2001). This notion, along with the understanding that both uncertainties may exist concurrently, is part of the challenge in capturing and analyzing uncertainty. Epistemic uncertainty entails the ability to learn, increase knowledge, and reduce uncertainty. When epistemic uncertainty is about variability and uncertainty itself, it is referred to as "uncertainty about uncertainty", second-order uncertainty, or *ambiguity* (Einhorn & Hogarth, 1985).

We observe three different and independent uncertainty-related factors that have potential impact on the informativity of model statements: reliability, discovery, and simplification. Each one of these factors requires stochastic information value analysis. The information value of a random variable is calculated as its entropy, also known as the Shannon index: $H(X_i) = -\sum_{j=1,..,J_X} p_{ij} \log(p_{ij})$, where p_{ij} is the probability that $X_i = x_{ij}$ (Jaynes, 1957).

Reliability. The reliability-based informativity of model fact(i), is expressed by the Shannon entropy index, $EI_{reliable}(i)$ as defined in (1) as the entropy of the message based on its correctness, where

$p_{true}(i)$ is the probability that message(i) is correct or true. If the message correctness probability is 0.5, $EI_{Reliable}$(i) is maximized and is equal to *ln*(2). The entropy index has to be normalized according to its maximum value, to obtain the relative informativity figure (INF) in [0..1], $INF_{reliable}(i)$, as shown in (2). The INF of a message that has 0.5 probability of being correct – the ignorance prior – is 0, and the INF aspires to 1 when a message has probability 1 or 0 of being correct.

$$EI_{reliable}(\mathrm{i}) = -p_{true}(i) \cdot \ln\big(p_{true}(i)\big) - (1 - p_{true}(i)) \cdot \ln\big(1 - p_{true}(i)\big) \quad (1)$$

$$INF_{reliable}(i) = 1 - \frac{EI_{reliable}(\mathrm{i})}{\ln(2)} \quad (2)$$

Discovery. The discovery in a message is the extent to which the message provides new, previously unknown information, expressed as probability. More specifically, the message discovery is the probability that the message is already known to its receiver(s) prior to receiving it. If a message is unknown to all the receivers, the INF should be maximal, i.e. 1, while the INF of a definitely known (trivial) message is 0. The higher the probability that the message was known beforehand, the lower its INF is. The INF is therefore 1 when the probability that the message was previously known is 0, and 0 when that probability is 1. When that probability is 0.5, the INF is 0.5. $INF_{known}(i)$ is based on the entropic index $EI_{known}(i)$ as defined in (3).

An auxiliary index $V_{known}(i)$ defined in (4), follows the entropic index $EI_{known}(i)$ when the probability that message(*i*) is known is smaller than or equal to 0.5, and inverses the sign of the entropic index when the probability is equal to or greater than 0.5, expressing information value degradation. This function is continuous in $p_{known}(i) = 0.5$ where it is 0, and it monotonously decreases in [-1..1]. To restrict INF to [0..1], we project $V_{known}(i)$ from [-1..1] to [0..1] via linear transformation, yielding $INF_{Known}(i)$ in (5).

$$EI_{known}(\mathrm{i}) = -\big(p_{known}(i) \cdot \ln(p_{known}(i)\big) - \big(1 - p_{known}(i)\big) \cdot \ln(1 - p_{known}(i)\big) \quad (3)$$

$$V_{known}(i) = \begin{cases} 1 + \dfrac{EI_{known}(i)}{\ln(2)}, & p_{known}(i) \le 0.5 \\ \dfrac{EI_{known}(i)}{\ln(2)} - 1, & p_{known}(i) > 0.5 \end{cases} \quad (4)$$

$$INF_{known}(i) = 0.5(V_{Known}(i) + 1) \quad (5)$$

Simplification. A major purpose of conceptual models is to reduce complicatedness—the perceived complexity of the systems they specify. This purpose is delegated to the individual model facts. The INF of a model fact depends, among other factors, on its simplification potential—the extent to which the information reduces system complicatedness. The more a statement is likely to clarify, simplify, or reduce complexity, the more informative it is. The more the statement is likely to complicate matters (increase complicatedness), the less informative it becomes. The INF of a statement is defined in [-1..1]. If the statement is more likely to be complicated and therefore confuse the reader more than help him or her understand the system, then its INF is negative. The entropy of simplification by message(*i*), $EI_{simp}(i)$, is defined in (6) based on the simplification probability $p_{simp}(i)$. The relative entropy $V_{simp}(i)$ is defined in (7). $INF_{simp}(i)$, the INF for simplification is defined in (8). It negates the relative entropy when $p_{simp}(i) \le 0.5$, and matches it when $p_{simp}(i) \ge 0.5$.

Figure 1 depicts the informativity functions of the reliability, discovery, and simplification factors, as it depends on the probabilities p_{true}, p_{known}, and p_{simp}.

$$EI_{simp}(i) = -\left[\left(p_{simp}(i) \cdot \ln(p_{simp}\right) + \left(1 - p_{simp}\right) \cdot \ln(1 - p_{simp})\right] \quad (6)$$

$$V_{Simp}(i) = 1 - \frac{EI_{simp}(i)}{\ln(2)} \quad (7)$$

$$INF_{simp}(i) = \begin{cases} -V_{simp}(i), & p_{simp}(i) \leq 0.5 \\ V_{simp}(i), & p_{simp}(i) > 0.5 \end{cases} \quad (8)$$

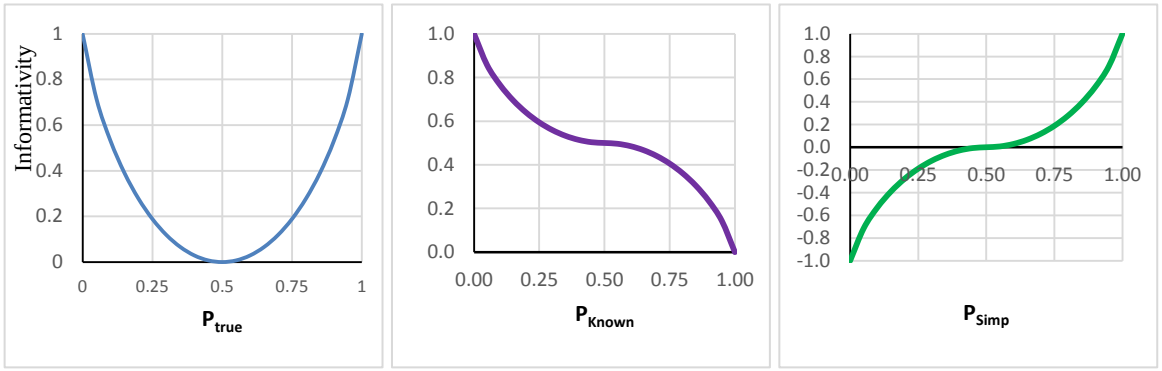


Figure 1. Informativity functions of specification uncertainty factors

Meta-Specification

Meta-specification is the specification of information about the specification, providing such details as the specification's maturity level, the rationale for its existence, its originator or originating source, the category it pertains to, and its implementation or otherwise-promoting priority.

Some meta-specification factors have been discussed as attributes of system requirements (Hull, Jackson, & Dick, 2006). Requirements are a subset of specifications: they are requested, desired, or expected specifications for systems to be developed. Specifications include, in addition to requirements, various constraints, conditions, concepts, design variables, and environmental contexts. Requirements engineering is a phase in the specification lifecycle. Requirements – especially customer-defined ones – are often external to the model, whereas the model, in addition to representing the requirements in the problem domain, is expected to demonstrate that the system satisfies the requirement set in the solution domain by deploying a certain concept, usually after considering several alternative ones.

Requirements modeling and model-based requirements engineering, as well as model-based requirements-specifications integration, have been defined as major research directions (Cheng & Atlee, 2007). OPM advocates starting with modeling requirements with the stakeholders as a first stage in a model-based lifecycle-encompassing approach. The model then gradually evolves from a problem-oriented, requirements-based context towards conceptual, solution-oriented architecture and design, which satisfy the requirements based on a concept that had ideally been selected as the best among several competing concepts according to predefined criteria.

Meta-specification factors contribute to the informativity of the model. However, they are usually not explicitly specified as part of the formal specifications. Adopting a holistic lifecycle approach, OPM encourages expressing meta-specifications as part of the model. However, this approach may be difficult to follow, especially in early conceptual modeling phases, when the system's lifecycle is still vague and fuzzy. It is therefore necessary to list the meta-specification factors that contribute to the informativity of individual model facts, and propose ways to incorporate them into the formal OPL

textual specification, such that they generate valuable information on the one hand, and preserve the text clear, lean, formal, and usable on the other hand. We elaborate on each one of the maturity, rationale, originator, category, and priority meta-specification attributes.

Maturity. The maturity of a statement is the lifecycle phase of its realization. Statements can describe any concept from its initiation and inception through its design, implementation, testing, and deployment, to current state-of-affairs, operation and maintenance, or retirement. The information about the maturity of the statement helps system stakeholders identify influence areas and decision points in the system's lifecycle. It also caters to retrieving statements based on their maturity level, for example in order to allocate functional concept statements to detailed design and well-designed solution statements to implementation. Practitioners often encounter or use the word *maturity* from their own role-centric perspective. Designers refer to design artifacts as "mature" when they are considered ready for development or production. Developers refer to functionality as mature when it reaches a certain level of wholeness or some peak of expectations. Operators refer to systems as "mature" when they reach a certain degree of operational capability. While all these perspectives on maturity are local, they merely reflect local gateways or entry or exit criteria, which are all part of a long, evolving thread.

It is good practice to define a finite and manageable set of maturity levels that best fit the needs of the project or system of interest. Assigning a maturity level to each model fact is a challenge, since statements tend to mature over time, which mandates constant model and statement maturity maintenance as part of ongoing model validity preservation. Sample informativity values of model facts due to their maturity level is proposed in Table 2. Each maturity level has a recommended or default informativity figure, which is based on the extent to which model facts associated with this maturity level can be influential in the model-based systems engineering process. This value can be changed according to technical or programmatic preferences and priorities. In this scheme, the INF of a model fact increases gradually towards the allocation phase – just prior to the beginning of implementation or development – when it is the most critical opportunity to influence or validate the statement. As soon as implementation begins, the capability to react or interfere with the realization of the statement gradually decreases. A fully operational concept maturity specification is the least informative, because almost nothing can be done to alter it at this stage.

Table 2. Possible informativity values of model facts due to their maturity level

Precedence	Maturity Level	Explanation	INF
1	Initiation	Coming of idea into existence	0.6
2	Conception	Creating a systems concept	0.7
3	Elaboration	Detailing the design	0.9
4	Allocation	Assigning or posting for implementation	1.0
5	Implementation	Developing or prototyping	0.8
5	Verification	Testing and evaluation	0.7
6	Production	Manufacturing or integrating	0.6
7	Introduction	Marketing deploying, or driving adoption	0.5
8	Operation & Maintenance	Using and maintaining	0.4
9	Retirement	Phasing the system out	0.1
99	Not specified		0.0

Rationale. The rationalization, justification, and motivation of design decisions is a challenge in various engineering disciplines, including mechanical, electrical, software, and systems engineering, whose importance in modern environments gains slowly-growing recognition (Dutoit, McCall, Mistrik, & Paech, 2006; Dutoit & Peach, 2001; Klein, 1993). Model and design statements constitute the

"what"—the structural system aspect, and "how"—the behavioral system aspect, while rationale constitutes the "why"—the functional system aspect, which concerns the benefit the system provides. Rationale management tasks include rationale elicitation, representation, communication, association, application, tracking, and validation.

Rationale can be represented in natural language, as rules in a knowledge-based system, or as an argument structuring graph. With OPM, rationale can be captured in several ways. First, it can be captured in free language as part of model element (thing or link) documentation. Second, it can be modeled as part of the system model and associated with model artifacts, which makes it integrated into the model, but more difficult to isolate and distinguish semantically from functional or design statements, unless clearly indicated as pertaining to some rationale class. Representing rationale in the model requires additional modeling effort and involves perception modeling, which system engineers are not accustomed to. UML and SysML allow analysts to attach rationale notes to various model elements, and thus provide clarifications and justifications in free text. Rationale is hence more critical when a statement's maturity and simplification potential are lower. It is important to rationalize early and preliminary design decisions and peculiar statements. It is also important to rationalize current state specifications when these are difficult to comprehend.

Originator. The origin or originator of a statement is the group, person, or role that are responsible for defining, requesting, requiring, or capturing it. By default, we may assume that the origin of all the statements in the model is the client or the chief system architect or the top-user or someone else who is the owner of the system model or of the requirements specification document. Specifying the originator has several benefits. First, it associates the statement with a particular owner, who is the "mother and father" of the statement and can provide answers, clarifications, or justification for that statement. Second, each origin can be assigned an authoritative index that affects the informative value of the statement. For instance, when a system is developed for a specific customer or client who has the budget and expects to use the system, that client's voice is more significant than, say, a junior software architect's voice, as the latter may be more interested in using novel technologies than satisfying the customer's requirements. Conversely, a technically-oriented architect would be much more significant than a client or end user when the product is an infrastructure system that interacts heavily and intimately with networking, communication, and processing units. Finally, a novel proof of concept for a breakthrough technological or scientific discovery, such as nanotechnology or molecular biology, would require the voice of the principal researcher or scientific domain expert to be heard first and above all the others.

Category. The category of a statement is the classification of that statement based on the message that it conveys. Statements can pertain to various problem-specific categories: contractual commitments, functional or non-functional requirements, natural or artificial facts and constraints, architectural considerations, risks, legal, regulatory, or procedural constraints, usability, etc. It is good practice to define a finite and manageable set of categories that best match the project or system of interest. Each statement should be associated with a category.

Priority. The priority of a statement is the importance of promoting it. It is good practice to define an agreed-upon scale of priorities (e.g., "critical", "mandatory", "important", and "nice to have"). Not every statement needs prioritizing. Some statements describe current state or natural facts, for which priority definition is meaningless or redundant.

Priority is usually specified in an increasing order: the lower the value of the priority index, the higher the importance. In some places Priority “Zero” is used for mandatory or critical items that the system of

interest must have. Priority specification has to be clearly defined, well-graded, valid, and reliable. The priority specification informativity index has to guarantee that even the lowest priority level has some positive information value, and that the value of each priority is based on the number of priority levels, on the proportion of items in each priority level, and on the expected or intended number or proportion of items in each priority level, according to the prioritization scheme.

A good prioritization scheme should balance the number of items in each priority level. There are two approaches – uniform and pyramidal. The uniform prioritization approach requires the number of items in each priority level to be about the same. If the number of priority levels is $N_{Priorities}$, then the number of items in each priority level should be $N_{Statements}/N_{Priorities}$. This approach is useful for workload balancing, as it allows for matching the item feed ratio to resource capacity. This scheme enables the number of development iterations to be determined by the number of priority levels or vice versa.

The pyramidal approach caters to managerial attention distribution rather than workload balancing. The uniform approach makes it more difficult to discern the few most important items that require most of the managerial attention. In this approach, the number of items in priority level k, $k=0,..,$ $N_{Priorities}$, should be $(k+1)/(S_{Priorities} + N_{Priorities})$where $S_{\text{Priorities}} = 0.5 \cdot N_{Priorities}(N_{Priorities} + 1)$, i.e., the sum of priority indices. Some combination of the uniform and pyramidal approaches might benefit from the advantages of both: loading the balance while catering to effective program management.

Specification Management

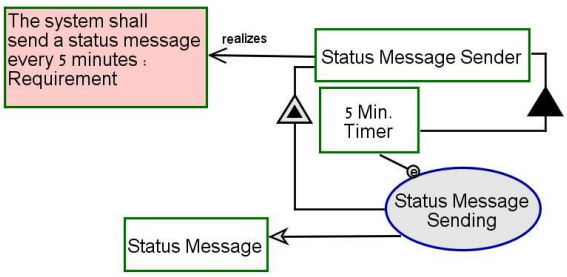
In this section we discuss traceability and demonstrability—two aspects associated with specification lifecycle management that contribute to the informativity of the specification.

Traceability. Complex systems are built to meet a set of requirements. Towards system delivery or product release to market, the systems engineering group is required to demonstrate how each requirement can be traced to a set of realization factors – structures, mechanisms, functions, performance criteria, in-system and on-system procedures, etc. This process is meant to provide evidence that the system satisfies the requirements of the stakeholders and customers. Likewise, design factors have to be traced back to operational, regulatory, functional, architectural, technical, or non-functional requirements, which may have resulted from higher-level requirements or constraints. Eventually, each requirement specification has to be traced to a design specification, and vice versa.

Traceability assurance is a wearisome task that often requires reviewing hundreds and thousands of specification pages and tracing each line to a source of authority. MBSE can potentially reduce some of the cognitive load, effort, and error rate associated with this process by managing traceability among various model artifacts. The problem with UML and SysML is that the number of symbols or syntactic building blocks is well over 100, which means that the number of possible traceability link types is unmanageable. OPM has only two model element kinds: things and links. Things can only be processes or (stateful or stateless) objects. Links can be structural or procedural. Syntactically, this is the lowest level of detail required to manage traceability to requirements, comprising at most 6 or 7 kinds of traceable elements. Various kinds of links constitute semantic patterns rather than syntactic building blocks. With this in mind, it is much easier to trace model elements to requirements of all sorts.

OPM models can also include raw requirements by defining them as conceptual objects in the model. Tracing model elements—processes, objects, and links—to those requirements is as easy as connecting two objects with a tagged structural link carrying the default tag "relates to" or a more meaningful "traces to" or "realizes" tag from the system artifact to the requirement. OPM mandates that neither

object nor process be represented in the model without being linked to at least one other thing (object or process). Any requirement object that is not connected to any other object that satisfies it can thus be easily pinpointed. In this case, requirement traceability is evaluated for informativity just like any other tagged structural link according to Table 1. An example of requirements management and traceability that are built into the model is shown in Figure 2. This simple model shows that the object "**Status Message Sender**" realizes the requirement "**The system shall send a status message every 5 minutes**". The Object-Process Diagram (OPD) on the left is equivalent to the Object-Process Language (OPL) text on the right.



1. **The system shall send a status message every 5 minutes** is of type Requirement.
2. **Status Message Sender** exhibits **Status Message Sending**.
3. **Status Message Sending** requires **5 Min. Timer**.
4. **Status Message Sending** yields **Status Message**.
5. **Status Message Sender** consists of **5 Min. Timer**.
6. **5 Min. Timer** triggers **Status Message Sending**.
7. **Status Message Sender** realizes **The system shall send a status message every 5 minutes**.

Figure 2. Built-in Requirements Traceability in an OPM Model

Demonstrability. There is a significant advantage to the ability to show something working, be it only hypothetically. The ability to execute, simulate, or demonstrate a statement in action in a virtual or actual environment has many benefits, as it provides for verification, validation, and illustration of the statement at hand and of the model of the system as a whole (Bolshchikov, Somekh, Mazor, Wengrowicz, & Dori, 2015). Statements differ in demonstrability. Simulating complex dynamic behavior is often more challenging than visualizing the structure of a part, even in three dimensions.

There are several levels of demonstrability, each providing a different value. The highest value can be gained from demonstrating the statement in a working system. This is supposed to be trivial for statements that are defined as pertaining to the "current state" of the system. Some statements cannot be demonstrated with the actual system, because such demonstration might result in significant damage to or by the system. For instance, demonstrating the ejection of a pilot from a fighter jet is not something we can demonstrate just for educational or show purposes, because this would destroy the aircraft, incurring a prohibitive cost and endangering the life of the pilot. Such a scenario can be tested in an experiment or on a physical testbed, but a test like this will never match the real thing in its ability to conclude that the statement is completely valid and verifiable. A physical or conceptual model of the system can be used to test various structural and behavioral configurations. A fully-executable model – at any level of completeness – provides the respective INF to all the model facts.

The Integrated Informativity Index

Having specified and discussed a variety of informativity-enhancing factors (IEFs), we now aggregate the informativity figure (INF) generated due to these IEFs. The IEFs are clustered, listed and defined in Table 3, showing that each IEF has a different level of importance and criticality, expressed as a relative or numerical weight.

Each IEF[j], j=1,..,J, has an informativity function, which maps possible IEF values to INF values in [0..1] or [-1..1]. Each IEF has a relative weight w_j. The INFs of the individual specification "messages" m_i are independent. Using a linear weighting scheme shown in (9), the INFs of all the individual messages $i = 1, .., N_{Specs}$, are summed into the weighted INF, $WINF(i)$. Similarly, the total INF of each

IEF over all specification messages, $FINF_j$, is defined in (10). The model's integrated informativity index, I^3, is the sum of the *WINF*s of all specification messages, and also the weighted sum of all $FINF_j$, as defined in (11). I^3 is an unbounded index. As models evolve and change constantly, the significant measure is the change in $I^3(v_{new})$ relative to $I^3(v_{old})$, denoted and calculated in (12) as $\Delta I^3(v_{new}, v_{old})$, where v_{old} and v_{new} are the model's earlier and later version indices, respectively. $FINF_j$ can also be considered separately without weighting and aggregating them, as determining a weighting scheme may not be trivial or straightforward. Partial aggregation, e.g., at the IEF cluster level, is also an option.

Table 3. Informativity-Enhancing Factors (IEFs) with possible weights

IEF Cluster	IEF name	Weight	IEF Definition
Specification (e.g., 40%)	Specification Pattern	40	kind of model fact, such as thing definition or link, as specified in Table 1, including possible extensions such as affiliation or essence specifying as part of thing definition, state specifying as part of structural and procedural links, and path and logic specifying as part of procedural links.
Uncertainty (e.g., 30%)	Reliability	12	probability of correctness or truthfulness of the statement
	Discovery	12	subjectively-defined probability that a model-fact was previously unknown to model viewers.
	Simplification	6	extent to which the statement clarifies, simplifies or reduces the complexity of the system aspect it describes.
Meta-Specification (e.g., 20%)	Rationale	5	reasonable justification for introducing a statement in a model
	Initiator	5	stakeholder who initiated the statement, which can be client, beneficiary, owner, user, product manager, architect, designer, analyst, developer, tester, etc.
	Category	5	classification of the statement according to its programmatic or technical implication.
	Priority	5	urgency of or preference for developing or implementing the statement, which can be mandatory, desirable, optional, etc.
	Maturity	5	lifecycle stage of the statement, such as design, implementation, testing, production, and operation.
Model Management (e.g., 10%)	Traceability	3	ability to trace design specifications to requirements.
	Demonstrability	7	ability to present, demonstrate, simulate, or execute the statement in a simulated or real environment.

$$WINF(i) \equiv \frac{\sum_{j=1}^{J}\left(w_j \cdot INF_j(i)\right)}{\sum_{j=1}^{J}(w_j)}, i = 1,..,N_{Specs} \quad (9)$$

$$FINF_j \equiv \sum_{i=1}^{N_{Specs}}\left(INF_j(i)\right), j = 1,..,J \quad (10)$$

$$I^3 \equiv \sum_{i=1}^{N_{Specs}}\left(WINF(i)\right) = \frac{\sum_{j=1}^{J}(w_j \cdot FINF_j)}{\sum_{j=1}^{J}(w_j)} \quad (11)$$

$$\Delta I^3(v_{new}, v_{old}) = \frac{I^3(v_{new})}{I^3(v_{old})} - 1 \quad (12)$$

Discussion and Conclusion

Model informativity is a prime indicator of the model's usefulness. While this notion is grounded in both information theory and utility theory, studies of conceptual model informativity evaluation have been scarce, as opposed to stochastic models. Quantifying, measuring, and enhancing conceptual system model informativity provide means to constantly enhance the model's usefulness for its stakeholders. Motivated by this understanding, we have presented MIA – a framework for model informativity analysis – MIA. In addition to being a normative approach for measuring and evaluating

model informativity, MIA is also a constructive approach, as it points out model areas and aspects in which informativity is low, possibly directing modeling efforts towards improving them.

We have defined a set of informativity-enhancing factors (IEFs) – attributes of formal conceptual model specifications that facilitate information extraction from models. The specification pattern is an essential and significant IEF. We have demonstrated how a model fact can be evaluated based on its specification pattern's importance and detail-extensibility. A second group of IEFs adds a stochastic dimension to specifications by relaxing deterministic assumptions on the reliability, discovery, and simplification of the model by each model statement. We use Shannon entropy analysis to infer the effect of specification uncertainty, as each statement is considered a message in a communication session (the model) – on each of these IEFs. A third group of IEFs includes meta-specifications: various specifications' attributes that are not explicit in the actual statements but characterize them. These include model maturity, rationale, initiator, category, and priority. Model stakeholders can determine INFs for these attributes based on their subjective priorities. Some meta-specifications can be integrated into the specifications they characterize in order to foster the systems engineers' awareness of the need to populate them with meaningful values. This is a matter for future research. The last group of IEFs, which concerns model and specification management, includes traceability and demonstrability. OPM accommodates requirements traceability as part of the model, rather than via detached mapping. OPM's simulation engine provides a good basis for model demonstrability, rendering all executable model-facts demonstrable at least by simulation if not by actual operation.

MIA is primarily a subjective, heuristic approach, in which values, parameters, weights, and likelihoods are estimated subjectively by stakeholders, and the aggregate measures are heuristic and utilitarian. This does not undermine the validity of MIA, as the analytical foundations of subjective judgement, utility, information, and probability are well-defined (Cooke, 1991; Pratt, Raiffa, & Schlaifer, 1964; Savage, 1972). MIA's advantage in this aspect is its ability to accommodate a variety of subjective informativity measures. It allows for eliminating irrelevant or inapplicable informativity measures and adding domain-specific factors or ones that may arise in future research and application of this approach.

Due to the limited scope of this paper, we could not elaborate on evaluation and analysis considerations for each and every IEF. This is left for future research. We intend to demonstrate this framework on actual models and show how their informativity improves as they evolve as more aspects and details are added to the model. Progress is underway in the area of knowledge-based design automation. Such models would enable producing quantitative evidence of the significance of particular modeling and design patterns integrated into the model and the extent to which they improve model informativity. We also study the improvement in informativity of protocol and standard models as part of transitioning to model-based standard and protocol authoring, which is expected to significantly increase their formality, consistency, and validity. Finally, we have recently begun an empirical research on informativity analysis in several industry projects, including the communication of the results to participants and the study of the impact on model evolution and on the modeling effort. This research will validate the claim that model informativity analysis is both reflective of model utility and potentially constructive for modeling efforts when communicated to the modeling teams.

Acknowledgment

We thank Gordon Center for Systems Engineering at the Technion for funding this research. We also wish to thank the anonymous INCOSE IS reviewers for their useful comments and suggestions.

References

Akaike, H. (1974). A New Look at the Statistical Model Identification. *Automatic Control, IEEE Transactions on*, *19*, 716–723.

Azrieli, Y., & Lehrer, E. (2008). The value of a stochastic information structure. *Games and Economic Behavior*, *63*, 679–693.

Bedford, T., & Cooke, R. (2001). *Probabilistic risk analysis: foundations and methods*. Cambridge University Press.

Bernardo, J. M. (1979). Expected Information as Expected Utility. *The Annals of Statistics*, *7*, 686–690.

Bolshchikov, S., Somekh, J., Mazor, S., Wengrowicz, N., & Dori, D. (2015). Visualizing the Dynamics of Conceptual Behavior Models: The Vivid OPM Scene Player. *Systems Engineering, In Press*. doi:10.1002/sys.21321

Bowdle, B. F., & Gentner, D. (1997). Informativity and asymmetry in comparisons. *Cognitive Psychology*, *34*, 244–286.

Bozdogan, H. (2000). Akaike's Information Criterion and Recent Developments in Information Complexity. *Journal of Mathematical Psychology*, *44*, 62–91.

Cheng, B. H. C., & Atlee, J. M. (2007). Research Directions in Requirements Engineering. *Future of Software Engineering (FOSE '07)*. doi:10.1109/FOSE.2007.17

Cherubini, M., Venolia, G., Deline, R., & Ko, A. J. (2007). Let ' s Go to the Whiteboard: How and Why Software Developers Use Drawings. In *CHI 2007 - SIGCHI Conference on Human Factors in Computing Systems* (pp. 557–566). San Jose, CA: ACM.

Cooke, R. M. (1991). *Experts in uncertainty: opinion and subjective probability in science*. Oxford University Press, USA.

Dori, D. (2002). *Object-Process Methodology: A Holistic Systems Approach*. *Object Process Methodology - a holistic systems paradigm*. Berlin, Heidelberg, New York: Springer.

Dori, D. (2011). Object-Process Methodology for Structure-Behavior Codesign. In D. W. Embley & B. Thalheim (Eds.), *Handbook of Conceptual Modeling* (pp. 209–258). Berlin, Heidelberg: Springer Berlin Heidelberg.

Dori, D. (2016). *Model-Based Systems Engineering with OPM and SysML*. New York: Springer.

Dutoit, A. H., McCall, R., Mistrik, I., & Paech, B. (2006). *Rationale Management in Software Engineering*. Springer.

Dutoit, A. H., & Peach, B. (2001). Rationale management in software engineering. In *Handbook of Software Engineering and Knowledge Engineering*. World Scientific Publishing.

Dwyer, M. B., Avrunin, G. S., & Corbett, J. C. (1998). Property specification patterns for finite-state verification. *Proceedings of the Second Workshop on Formal Methods in Software Practice - FMSP '98*, 7–15.

Einhorn, H. J., & Hogarth, R. M. (1985). Ambiguity and Uncertainty in Probabilistic Inference. *Psychological Review*, *92*. doi:0033-295X/85/S00.75

Embley, D., & Thalheim, B. (2011). *Handbook of conceptual modeling: theory, practice, and research challenges*. (D. Embley & B. Thalheim, Eds.)*Vasa*. Springer. doi:10.1007/978-3-642-15865-0

Frankel, R., Kothari, S. P., & Weber, J. (2006). Determinants of the informativeness of analyst research. *Journal of Accounting and Economics*, *41*, 29–54.

Funes, A. M., & George, C. (2003). Formalizing UML Class Diagrams. In L. Favre (Ed.), *UML and the Unified ProcessProcess* (pp. 129–198). IRM Press.

Gnoevoi, A. V, & Chesnokov, V. M. (2003). Hierarchical Modeling of Flows of Media with Complex Rheologies in Channels and Recesses of the Working Members of Production Machines. *Chemical and Petroleum Engineering*, *39*, 450–456.

Goldstein, M., Tech, V., Va, B., & Rougier, J. (2008). Assessing Model Discrepancy Using a Multi-Model Ensemble. *Sciences-New York*, 1–35.

Grunske, L. (2008). Specification patterns for probabilistic quality properties. *2008 ACM/IEEE 30th International Conference on Software Engineering*, *61*. doi:10.1145/1368088.1368094

Howard, R. (1966). Information Value Theory. *Systems Science and Cybernetics, IEEE Transactions on*, *2*, 22–26.

Howard, R. (1968). The Foundations of Decision Analysis. *IEEE Transactions on Systems Science and Cybernetics*, *4*, 211–219.

Hull, E., Jackson, K., & Dick, J. (2006). *Requirements Engineering*. *Requirements Engineering* (Vol. 13). doi:10.1145/336512.336523

ISO/TC 184. ISO/PAS 19450:2015 Automation systems and integration — Object-Process Methodology (2015). ISO.

Jaynes, E. T. (1957). Information Theory and Statistical Mechanics. *Physical Review*, *106*, 620–630.

Klein, M. (1993). Capturing design rationale in concurrent engineering teams. *Computer*, *26*. doi:10.1109/2.179154

Konrad, S., & Cheng, B. H. C. (2005). Real-time Specification Patterns. In *Proceedings of the 27th International Conference on Software Engineering - ICSE '05* (pp. 372–381). St. Louis, MO, USA: ACM.

Maciejowski, J. M. (1979). Model discrimination using an algorithmic information criterion. *Automatica*, *15*, 579–593.

Pratt, J. W., Raiffa, H., & Schlaifer, R. (1964). The Foundations of Decision Under Uncertainty: An Elementary Exposition. *Journal of the American Statistical Association*, *59*, 353–375.

Reich, Y. (1995). Measuring the value of knowledge. *Int. J. Human-Computer Studies*, *42*, 3–30.

Reich, Y. (2002). General Design Theory as a Formal Theory of Design. In Amaresh Chakrabarti (Ed.), *Engineering Design Synthesis: Understanding, Approaches and Tools* (pp. 35–48). Springer-Verlag London.

Reich, Y., & Barai, S. V. (1999). Evaluating machine learning models for engineering problems, *13*, 257–272.

Rosenblueth, A., & Wiener, N. (1945). The role of models in science. *Philosophy of Science*, *12*, 316–321.

Savage, L. J. (1972). *The Foundations Of Statistics*. Dover Publications.

Trentelman, K. (2009). *Survey of Knowledge Representation and Reasoning Systems*.

Biography

Dr. Yaniv Mordecai holds a Ph.D. (2016) from the Technion – Israel Institute of Technology, Haifa, Israel; and M.Sc. (2010, cum laude) and B.Sc. (2002) in industrial engineering from Tel-Aviv University, Tel-Aviv, Israel. His research interests include model-based systems engineering, risk analysis, decision analysis, interoperable systems, and operations research. He is a proficient and active systems engineer, with expertise in aerospace and defense, information technology, command and control, and avionics systems. He has authored several papers on advanced model-based systems engineering problems, won several awards and grants, and acted as a manuscript reviewer for leading systems engineering journals and conferences.

Prof. Dov Dori, INCOSE Fellow, is the Harry Lebensfeld Chair of Industrial Engineering and Head of the Enterprise System Modeling Laboratory at the Faculty of Industrial Engineering and Management, Technion – Israel Institute of Technology, Haifa, Israel. He holds a Ph.D. (1988) in computer science from Weizmann Institute of Science, Rehovot, Israel; M.Sc. (1981) in operations research from Tel Aviv University, Tel Aviv, Israel; and B.Sc. (1975) in industrial engineering and management from Technion. His research interests include model-based systems engineering, conceptual modeling of complex systems, systems architecture and design, software and systems engineering, and systems biology. He is alternately Visiting Professor in Massachusetts Institute of Technology, Cambridge, MA, USA. He invented and developed object–process methodology, the ISO 19450 Standard. He has authored, co-authored or edited five books and over 300 journal papers, conference publications, and book chapters.