Measuring the Impact of Information on Complex Systems

Larry H. Reeker
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, Maryland
larry.reeker@nist.gov

Albert T. Jones
Manufacturing Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, Maryland
albert.jones@nist.gov

Abstract

The application of power-driven machinery to manufacturing and other areas of human endeavor characterized the Industrial Revolution in the 18th and 19th centuries. Measurement contributed in many ways to the increasing economic influence of these machines. Using formal or informal physical principles, metrics and measurement techniques were found that allowed the comparison of machine performance (evaluation), the development of machines with the needed qualities (engineering), and the coordination of machines within factories (integration). The required physical dimensions were space, time, and mass, and the common physical quantities derived from these three; and, for these quantities, measurement techniques were established. In the Information Revolution begun in the 20th Century, measuring information is also vital to the continued influence of machines. Unfortunately, information is not as well understood, as are physical constructs. It seems to have an unlimited number of dimensions, and no generally accepted metrics or measurement procedures. So how do we measure the impact of information in the 21st Century? This paper sketches research directions that may help to answer this question and it stresses the importance of obtaining an answer.

I. Introduction

The machines or systems (machine and system will be used as synonyms) to which we will refer in this paper are ones in which information is vitally necessary and for which information affects the behavior. Our use of "behavior" is not limited to input and output, not a black box definition. There is information coming into, going out of, and residing within a system that is essential to both its internal and external behavior. So machines have a physical aspect, but it is the informational one that will be stressed here. Of particular interest is manufacturing, where systems have practical importance plus high complexity; but the same problems arise in all information domains.

Information must be conveyed in physical symbols like marks on paper, sounds, or electrical pulses. Nevertheless, information has an effect on a system that is not explainable by its physical properties alone. That effect is related to (1) the *organization* of the symbols, (2) the *meaning* ascribed to the symbols and their organization, and (3) the change in the system *state* that comes from understanding and acting on that meaning. Since the state varies with time, so will the effect of a particular item of information on the properties of a given system.

There are three practical reasons for measuring these properties. First, they are useful to evaluate systems successfully. Measurements are needed to compare one system to another, to show that they meet a particular need, to prove that they fulfill the specifications of a prior agreement, to demonstrate that they conform to standards, and so on. Second, they are necessary to engineer systems successfully. Measurements are needed to ensure that constraints in the building process are met and that the system will behave in the required way. As an extension of the process of building to the practical need for modularity, they are required to integrate systems successfully. Measurements are needed to verify that the information needed by one system can be supplied by others without error and on time. The importance of measurement can thus be based in the three roles for measurement: *evaluation*, *engineering*, and *integration*. There are other reasons, too, that might be cited, such as *understanding* the system; but they can be seen generally as overlapping the three practical reasons[1].

In the earliest applications of information that impacted the performance of systems, the physical carrier of the information was mechanical links in steam engine governors, punched holes in Jacquard looms, or electrical connectivity in thermostats. The impacts of the information could be measured for these applications by its physical properties, and its physical cause and effect (as heat causing expansion of a certain amount or current

[1] An appendix is attached that discusses some meta-level aspects of the measurement, with respect to science and engineering.

flow in a thermocouple), reaction times, and so on. Those impacts could be quantified, therefore, in terms of system performance. The *meaning* of the information, not its *representation*, was what influenced that performance. The punched holes in the loom cards were originally in stiff pasteboard and were read by needles. After their evolution into Hollerith's paper cards, they could be read by pins that conveyed electricity and later by light and electricity. Finally, when the cards went away entirely in favor of other information representations, the same information could be conveyed by different physical means. Thus, the performance of many physical systems is, in some sense, independent of the physical form of the information that drives them.

The complexity of systems has evolved considerably over the past twenty years. Among the more complex systems are what we often call "intelligent systems". In these systems,[2] the impact of the informational component, its representation and its meaning, is paramount. It is clear that testing for the amount of and the impact of information in any particular area is going to be difficult, and that even the terms "amount" and "impact" will be difficult to define. In short, a metric of the information abstracted from the physical parameters is not evident. The paper will argue that a great number of such systems exist, even outside the area that might be labeled "intelligent" or "knowledge based". It stresses several critical points to understanding and controlling such systems.

II. Impact of Information on Control of Complex Systems

Two of the simple systems mentioned above as examples – the thermostat and the governor – are ones in which the information is gathered by feedback, which is the collection of information, its representation in a physical medium, and its interpretation to control a system. The handling of information for control can be much less direct. Yet it is often the case that simple models can provide ideas that can be generalized to more complex ones, and maybe it can help in this case to understand the general problem of measuring information impact.

Many complex systems can be viewed as a collection of integrated and layered subsystems, which might at some "bottom" layer be cases of direct physical control. Typically, the layering occurs in both the temporal domain and the spatial domain. The bottom layer contains some combination of biological, chemical, and physical processes. In humans, the evolution of these processes is governed by an internal and natural intelligence, which we call the mind. While we do not know exactly how it works, we see its benefits every moment of our lives.

Man-made systems, on the other hand, are not endowed with a mind. The processes that make up these systems are subject to the second law of thermodynamics. Hence, without any external intelligence to guide their evolution, entropy will increase and they will go out of control over time. To keep this from happening, researchers have expended an enormous amount of time, energy, and money to develop models, algorithms, and heuristics that come under the general heading of control theory.

While it is conceptually simple, control theory can be complicated in practice. Conceptually, it consists of two steps. Step 1 is to set the desired goal and develop a plan to achieve that goal. Step 2 is to observe the execution of that plan and make adjustments as required. The first step usually involves the development of a model of the system, an optimization problem based on that model, and technique to solve that problem. Models, which can be continuous or discrete, and deterministic or stochastic, typically have temporal and spatial parameters. The optimization problem has at least one measurable, quantitative goal and constrains the parameters in the model. Sometimes these problems can be solved analytically, sometimes not. Regardless of how the solution is derived, it results in a plan to be executed by the system.

Consider a robot that that must move a part from point A to point B in the shortest possible time. To generate a path to accomplish this goal, the robot controller, which could be a human or a software procedure, needs models of the robot and its environment. These models are continuous time, continuous state, and deterministic. The controller formulates an optimization problem whose solution will specify the start coordinates, the end coordinates, the time, limits on models parameters (such as speed, joint angles, and so on), and possible obstacles to avoid. That solution yields the optimal plan that the robot should use. This plan is then sent to the robot, or more accurately the execution part of its controller, to be implemented. Once the robot begins to move, we must proceed to step 2. This means that we must somehow make sure that the robot does not exceed any of the limits and follows the predetermined path. We do this through the generation and analysis of feedback. Sensors on the robot create the feedback, which is analyzed by the controller. When a problem is detected, a new plan will be generated.

CRITICAL POINT 1: Both the plan and the feedback are information objects, which impact the performance and the behavior of the robot. Some of these objects are simple; some are not. The meaning of these objects must be conveyed to and understood by

[2] This paper used the term "complex" interchangeably with "intelligent", to avoid defining the latter term, the difference not being important for our purposes.

all hardware and software components or there is no hope of achieving the desired goals. These capabilities do not happen "naturally"; they must be built into the system.

As we move up the layers, we no longer deal directly with biological, chemical, and physical systems. Instead, we deal with decision-making and information systems that affect those bottom-layers, but on a longer-term basis. Nevertheless, the same two steps are involved. In this case, however, the models are discrete time and discrete state systems that often contain one or more stochastic parameters. There are several, often conflicting, quantitative performance measures and the techniques are implemented in a number of software applications such as linear programming, demand forecasting, and supply chain management. These applications also produce plans that are implemented in other, lower-layer software applications -- demands lead to production plans, which lead to schedules, which lead to sequences and so on. These plans are based on information that has a high degree of uncertainty. Some of this uncertainty arises because of the influence of the second law on the bottom-layer processes. Some of it arises because of the stochastic nature of predictions associated with demand projections, priority orders, and material arrivals, to name a few.

CRITICAL POINT 2: Optimizing high-level performance measures is critically dependent on the ability of the associated software applications to share complex information objects. Furthermore, without have a common understanding of the meaning of those objects, optimization is useless.

As we progress through the various layers of a complex system like a manufacturing enterprise, an evolution occurs from continuous time to discrete time and from continuous state to discrete state. Furthermore, an aggregation in information takes place as well - very detailed, relatively simple, deterministic information at the bottom; very little detail, more complex, highly stochastic information at the top. No one knows how this evolution or aggregation takes place. Moreover, at every layer, there is some influence of entropy from both the second law and information uncertainty. At the bottom, the second law dominates. At the top, information uncertainty dominates. We have a very good idea of how to measure and control the effects of the second law on physical system performance. We have almost no idea how to measure and control the effects of information on performance.

CRITICAL POINT 3: Information has a large impact on system performance. Integration, getting the right information from one software application to another, also has an impact. Consequently, ensuring that all software applications have the same understanding of that information is critical to system performance. Furthermore, and most importantly, our ability to measure how well they understand impacts directly our ability to measure the true performance of the system.

An important question then is how can we build software applications that are capable of understanding information. The simple answer is that we must make software, just as we must make equipment, more intelligent. More accurately, we must surround each software application with the "stuff" it needs to understand the information it receives from other applications. A partial list of some of that “stuff" includes:

Parsers to determine the structure of an encoding (the physical representation) according to a known structural description (for symbols, called a “grammar”).

Ontologies to describe the internal model that the system can use to recognize inputs in terms of catalogues of entities and processes and their relationships.

Dictionaries to define the relationship of discrete elements of the encoding to objects and processes in the ontology.

Mappers from encodings to models or directly from one model to another.

Controller which makes decisions on a course of action (a sequence of behaviors), based on information in plans that have been preprogrammed or formulated and inputs (from users or sensors, including feedback), and operates actuators to cause the behavior sequence.

Actuators: Devices which behave physically to produce behavior.

Perceptors: Systems that convert the input of sensors into information for the system to process.

Equivalence, Similarity, and Difference Metrics: Ways of measuring how the information in one system or subsystem relates to another – whether it is equivalent or not (more on this below!)

CRITICAL POINT 4: Our ability to control the performance of the physical systems can depend directly on our ability to measure the similarities and differences between information objects.

III. Measuring Equivalence between Information Objects

Perhaps the first thing to consider in looking at measurement metrics is whether definitions of equivalence can be established. This is a tricky issue, because in some sense they cannot. Consider two ontologies, as defined above. They conventionally are represented by classes of entities and their attributes, linked into hierarchies (lattices are mathematically one representation) based on the IS-A relationship. IS-A relationships are based on the attributes of classes of entities, and those attributes are based two things: fundamental properties and the behaviors of entities in activities. Trying to compare behaviors of entities after a certain degree of complexity is reached leads to things like the halting problem. Thus, just as it may be formally undecidable if two programs are equivalent, it may be difficult to determine ontological equivalence formally. Perhaps we can still get measurements that will enable satisfactory performance within bounds, and undecidability will not be a problem. We still need to measure some concepts of equivalence, even with the blanket restriction of undecidability, which is a common restriction that must be sidestepped often in computing. One approach is to use approximations, which are often required by limited measurement precision anyway.

CRITICAL POINT 5: The equivalence of information objects may be undecidable, but we may be able to develop approximate measurements.

Developing an approximate equivalence metric puts us right in the middle of an ongoing controversy. That controversy revolves around the best way to represent uncertainty in information. There are two views: probabilistic and fuzzy. The probability proponents argue that there is only one consistent way to measure uncertainty and that is probability theory. They further argue that all probability is conditioned upon prior information and that the proper way to do inferencing must be based on a Bayesian framework. That framework says (1) create a prior distribution using the Principle of Maximum Entropy, (2) update that distribution using any new information and Bayes theorem, and, (3) use this new distribution for inferencing [Jaynes, 88].

The fuzzy proponents argue that information is not crisp enough to be measured using the quantitative laws of probability. To overcome this difficulty, the concept of a membership function is used. It has yet to be determined for many researchers if there is any essential difference between using fuzzy information and exact numbers with probabilistic error bounds. At this point, many people agree that fuzzy information can be a useful concept for engineering systems and simplifying the code that runs those systems. It may turn out that it is a mathematical difference analogous to that between matrix and wave mechanics in physics.

CRITICAL POINT 6: A full understanding of the relationship between various approximate ways of measuring information is needed.

Another important issue related to measuring uncertainty in information objects is the notion of entropy. That there is a relationship between information and entropy has been postulated for many years. A number of information measures have been proposed [Arndt, 01], including those by Shannon [Shannon and Weaver, 71] and Stonier [Stonier, 91]. Information is a measure of the decrease of uncertainty, and its representation requires an organized notation. Entropy is a measure of the increase of randomness. If one takes an organized body of information and randomizes it (adds noise) then there is less information and higher entropy.

The term “information” is itself used in different ways, however, because organization can mean many things. In thermodynamics, it is molecules behaving in an organized fashion. In Shannon’s communication examples, it is strings of symbols sent from a sender arranged in a way that can potentially lessen uncertainty at a receiver that can decode the symbols. In other uses, however, information has to be relevant to some task being performed by a system. In computation, it is related to complexity considerations. The work of Solomonoff, Kolmogoroff and Chaitin [Chaitin, 92] links information conveyed by symbols in logic and information systems with complexity of computation, and relates them to Shannon’s measures, as well.

The problem of information content is that it is “about *something*”. How do we compare information about two different subjects? The answer may be that we just do not do so, at least if the subjects are independent. But how do we know if they are independent? We may not want to mix oranges and apples; but, if we are concerned about fruit, we can develop information about them because they are no longer independent. Consider the following simple experiment. Suppose we have nine pieces of fruit, five oranges and four apples, and someone puts three of them in a bag. If we find three apples, we know that there are no oranges. This becomes much more difficult when we get to questionable or fuzzy sets -- try repeating this experiment with five big apples and four small apples. This second experiment is typical of problem of measuring information content. It depends on the individual system and its ontology. Independence can be classified as being in different dimensions, analogous to dimensions in physics; but it seems there are too many dimensions to measure.

In the example above "fruitiness" might be considered an attribute and the question might be whether a tomato has some fruitiness, so a negotiation is needed to decide if it will count or not. The Garden of Eden "fruit" is generally considered to be an apple. Could it be an orange? Is an apple "fruitier" or more likely to be fruity than an orange? Reasoning like this would call for a lot of dimensions, since apples and oranges alone have plenty of attributes to be compared. The psychologist and communication scholar Charles E. Osgood developed work in the measurement of a type of meaning (which is information content in much the same way that work is energy; meaning changes information content).

Osgood was interested in connotative meaning – meaning that is related to an individual's personal ontology [Osgood, 57]. So, it is beyond the denotational meaning and only intended to be partial. In trying to define it, he postulated three dimension types or factors, within which pairs of adjectives would indicate denotations.

- Evaluative factor (example: good - bad)
- Potency factor (example: strong – weak)
- Activity factor (example: active - passive)

Osgood then measured each pair, for each factor, on a seven point Likert scale. In the apple example, perhaps "fruity" could be equated to one and "not fruity" could be equated to seven. He then constructed an n-dimensional space, n being the number of adjective pairs, for his "semantic differential".

Clearly, much more than the semantic differential is needed to do the evaluation that can lead to integration of several manufacturing systems or bioinformatics systems. However, Osgood's ideas fit into the idea of fuzzy frameworks, and it was an important step in trying to formalize the idea of how the vocabulary of humans may vary. Vocabulary, while not the same as ontology, is closely linked, and provides a way to get at human ontologies. With machines where we know the code, we have the advantage of being able to read the ontologies more directly. The problem then becomes one of developing mappings from one ontology to another.

How do we reconcile the measures mentioned above with a system of dimensions like those used to measure physical dimensions, and how many dimensions do we actually have in information?

CRITICAL POINT 7: Even a theory that defines information not just as to amount but also in terms of "vectors" of information does not so far seem adequate for computing information equivalence or providing a precise measure of information overlap, though it is an interesting approach.

There are other problems of terminology that will not be discussed. It is rare to hear people use "data", "information", and "knowledge" in consistent, well-defined ways. And what about "potential information" that has a statistical amount but is not used at all? All of these still need some standard definition and scientific theories to put them in a framework. The underlying theory is not adequate. On the other hand, perhaps a limited categorization of knowledge that would cover particular programs is possible, if the categorization can be agreed upon. Here we return to the notion of ontology. This is a claim that sums up some of the ideas herein:

CRITICAL POINT 8: What we need to measure defines the model of the world that a system expects to find and its ways of coping with that world. The set of its behaviors may be infinite and unknowable if the machine is complex, but it is defined indirectly if we can predict behavior through the model. To predict behavior accurately, the information needs to be characterized in an ontology and what is done to information based on that ontology.

This point would seem to suggest an arduous task for satisfactory measurement of the relevant information that flows through a system; but it also suggest a potential benefit. Measurement of the information, if it is adequately powerful, can potentially "give back" to us in understanding enough value to repay the effort we put into creating and applying the metrics and techniques.

It is clear that every system that deals with information, whether computational or biological, contains an internal model of the outside world – an ontology. In computer programs, the elements of the ontology are data objects and procedures for manipulating those data objects; both are used by the program. Like matter and energy in physics, data objects and procedures in an ontology are related. Consider the notion of an ordered list of customers. It is a data object; yet it can be defined by a random set of customers and a sorting procedure. Its inputs are the (unordered) list and a statement of the type of order desired; and its output is the ordered list. If we want to integrate two systems that need an ordered list of customers for some purpose, it is important that we be confident that they employ the same order; otherwise they will not correctly operate together. To do that, it is necessary to measure the ontologies of the two systems to see if that is true.

The order example is not very complex on its surface. In practice, it is not possible, in general, to find out whether two algorithms producing an arbitrary order (not just a simple linear one) are outputting the same information. This is a consequence of a variety of undecidability results. So it is necessary to consider how we can use standards and measurements to be relatively confident

that there is going to be interoperability between the two systems.

IV. More on Measuring, Comparison, and Characterization of Ontologies

The point has been made above that a core ontology is needed to specify what information can be communicated to a given system by another system and what information the given system can send back. There is work going on in measuring these ontologies.

Every ontology has certain terms that may be grounded in physical parameters. In these cases, the physical information needs to be expressed in the appropriate dimensions. We are all familiar with the problem that arose in a probe of the planet Mars when the input and the expected physical parameters had different dimensions. That problem was not unique, and is even common in the building of software systems that do not have well-engineered descriptions of requirements. It is just easier to recognize when the information is closely related to physical parameters, as in the Mars probe. This also happens when the output from physical sensors is used as input to software applications. One tests the input requirements to see if the sensor outputs match exactly or in a way that is 'mappable' at the information level. Ontologies can help in both the matching and the mapping.

The development of explicit ontologies, therefore, is itself an important step because it clarifies which information items are directly grounded and which are indirectly grounded through computations. Comparing directly grounded objects is, in general, easier than comparing indeirectly grounded objects. Even if the frameworks for the ontologies are different, they can be compared if they use a consistent style. [Noy and Hafner, 97] characterized and compared a number of different ontologies. They concluded that if the ontologies can then be mapped into a similar format, they may be aligned, and maybe merged, with perhaps some human interaction. [Noy and Musen, 00] discusses this for a system called PROMPT; [McGuinness *et al*, 00] discusses an environment that provides tools for people who wish to merge ontologies.

Three efforts are underway to develop some standards for ontologies related to manufacturing: the Standard Upper Ontology, SUO, [http://suo.ieee.org/], the Process Specification Language, PSL, [http://www.mel.nist.gov/psl/], and, the Defense Agency Markup Language, DAML, [http://www.daml.org/] . These can be helpful in that they make the comparison of systems with different ontologies easier. How does each deviate from the core ontology? If the top (more general) ontological categories are the same, that saves a lot of work; and if they are not entirely the same, it may be easier to compare them to a single, core ontology and note their deviations. But there will still be systems with different ontologies in overlapping subject areas that need to be merged. A recent paper on how these may be compared is found in [Maedche and Staab, 01], who provide some explicit measures of similarity.

The goal of determining the properties of ontologies by analyzing the information in them and then comparing them to ontologies of other systems for interoperability purposes is still some distance away. Nevertheless, the interest in the area is growing and the results are promising.

V. Implications for Computer Science and Software Engineering

Each programming language must provide means of instructing a machine *how* to process information. This can be done implicitly, through logic or objects, or explicitly, through commands and procedure calls. Equally, a language must be able to convey knowledge of *what* information it is dealing with. This fact is encapsulated in the title of Nicklaus Wirth's book *Algorithms + Data Structures = Programs* [Wirth, 1976]. The history of programming languages shows that there is a tradeoff between describing the *how* and *what.* The tradeoff is illustrated by comparing object-oriented languages with procedural languages. In the SIMULA language, the first object-oriented language, is both procedural and object-oriented, but a glance at the programs in, say, *SIMULA Begin* [Birtwhistle *et al* 73] , illustrates the tradeoff.

The data structure is a fundamental part of a programming language. There has been descriptive work on data structures, and everybody has examples of "informationally equivalent" data structures. As a simple example, consider character strings. Before they were basic structures in some languages, they were handled as an array of characters and numbers with the number indicating the array address of the next character. Note that one of these has both single characters and integers, while the other one has only character strings. Despite the difference in structure, it is not difficult to show the informational equivalence of these two representations. In fact, the idea of *data abstraction* deals with using such equivalences to free the programmer from details of implementation. Though it is not the purpose of this paper to deal with programming *per se*, the idea of comparing ontologies is, in fact, similar to comparing two implementations that have different data structures. One first asks if the data structures are equivalent. If so, their *syntax*, the physical organization by which they are communicated by the programmer to the computer or stored in the computer is equivalent. The next question is "Do they mean the same thing?" Another way to ask this

question is “Are they informationally the same?” This is usually far more difficult to ascertain.

The data structures used in a program are a part of the program’s ontology, as are the procedures it uses. When we set out to integrate two existing programs, we want to be able to measure their ontologies because it is necessary that their data structures correspond in some way. That understanding allows us to couple them directly or through an interface. Two ways to improve software engineering are (1) to develop methods for creating and publishing these ontologies, and, (2) create processes for measuring informational objects and determining their role in the software programs.

One of the early issues in programming was modularity. As programs became more complex, and particularly as they began to be crafted by a team of people rather than a single individual, modularity became a design requirement. The possibility of reuse was another major impetus. Today, integration of software is probably more important than the creation of tailor-made programs. The challenge of integration is determining if the information-- data structures, knowledge bases and knowledge models, databases and their schemata, and the syntax and semantics -- are the same in each of the programs being integrated. Therefore, it is important to all systems that must share information – not merely the ones we might deem “intelligent” – that we have ways of measuring and comparing the information that each system uses.

VI. Conclusion

We do not know today how to measure equivalence of information nor its impact on a system. We need to be able to do so to evaluate existing systems, to engineer new systems, and, to integrate both. The benefits will be seen primarily in highly complex software systems that utilize a large amount of knowledge from either other programs with the system or devices in the real world. On the other hand, should the promise of “ubiquitous computing” come true, information will permeate physical systems as well. In this paper, we have argued that the measurement of the ontology of a system is a fundamental part of realizing these goals. We also indicated that some ideas are emerging in the areas of ontology standards and measurement.

References

Albus, J. S. [00] Features of Intelligence Required by Unmanned Ground Vehicles, in [Meystel and Messina, 00]

Arndt, C. [01] *Information Measures: Information and its Description in Science and Engineering*, Springer-Verlag, Berlin, Germany, 2001.

Birtwistle, G.M., Dahl, O-J., Myhrhaug, B., and Nygaard, K., *Simula BEGIN*,
Petrocelli/Charter, 1973.

Chaitin, G. J. [92] Information-Theoretic Incompleteness, World Scientific, 1992 (Reprinted 1998.

Chandrasekaran, B., J. R. Josephson, and V. R. Benjamins [98] Ontology of Tasks and Methods, 1998 Banff Knowledge Acquisition Workshop. [Revised Version is cited above. It appears as two papers "What are ontologies and why do we need them?," IEEE Intelligent Systems, Jan/Feb 1999, 14(1); pp. 20-26; "Ontology of Task and Methods," IEEE Intelligent Systems, May/June, 1999.]

Jaynes, E. 88, `The Relation of Bayesian and Maximum Entropy Methods (510Kb),' in *Maximum-Entropy and Bayesian Methods in Science and Engineering*, 1, G. J. Erickson and C. R. Smith (eds.), Kluwer, Dordrecht, 1988

Maedche, A. and S. Staab, Comparing Ontologies – Similarity Measures and a Comparison Study, Internal report 408, Institute AIFB, University of Karlsruhe, Germany, 2001.
http://ontobroker.semanticweb.org/ontos/report-aifb-408.pdf

McGuiness, D., R. Fikes, J. Rice, and S. Wilder, an environment for merging and testing large ontologies, *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*, Brekenridge, CO, 2000.

Meystel, A. and E. Messina, Eds., *Measuring Performance and Intelligence of Systems: Proceedings of the 2000 PerMIS Workshop*, National Institute of Standards and Technology, 2000.
http://www.isd.mel.nist.gov/research_areas/research_engineering/PerMIS_Workshop

Noy, N. F., and C. Hafner, 1997, The State of The Art in Ontology Design: A Survey and Comparative Review, *AI Magazine*, 18(3), 53-74 (1997)

Noy, N. F., and M. Musen, Prompt: Algorithm and tool for automating ontology merging and alignment, in *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, TX, 2000

Osgood, C. E., G. Suci and P. H. Tannenbaum [57*], The Measurement of*
Meaning. Urbana: The University of Illinois Press.

Reeker, L. [00] Theoretical Constructs and Measurement of Performance and Intelligence in Intelligent Systems, in [Meystel and Messina, 00]

Simon, H. A. [69] *The Sciences of the Artificial*. Third edition, Cambridge, MA, MIT Press, 1996. [First edition published 1969].

Shannon, C. and Weaver W. [71], *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, IL, 1971.

Stonier, T. [91], *Information and the Internal Structure of the Universe*, Springer-Verlag, Berlin, Germany, 1991.

Wirth, N. [76], *Algorithms + Data Structures = Programs*, Englewood Cliffs, N.J : Prentice-Hall, 1976.
Reeker, L., Theoretical constructs and measurement of performance and intelligence in intelligent systems, in [Meystel and Messina, 2000].

Walther, E., H. Eriksson, & M. A. Musen. [92] Plug-and-Play: Construction of Task-Specific Expert-System Shells Using Sharable Context Ontologies. AAAI Workshop on Knowledge Representation Aspects of Knowledge Acquisition, San Jose, CA, 191-198. AAAI, 1992.

Appendix: Technical and Scientific Progress Through Measurement

This appendix argues generally for the importance of measurement in technology and in science, of which the measurement discussed in the paper is an example. This importance is expressed succinctly in two statements of Lord Kelvin (William Thomson) in the 19th century. These statements are

"If you can not measure it, you can not improve it"
"To measure is to know."

The following sections describe in more detail what these statements have meant to engineering and science, thereby stressing the importance of paying attention of measurement considerations.

Let us assume that we are working on technology that is not underlain by an established scientific theory -- like robotics or AI. We should be aware of the relationship between technology and science, which is sometimes muddied when the public regards “information technology” and “computer science” as synonymous. Of course technology and science are linked, but they are separable, both logically and historically. As a rule, some technology in any given area has developed before the corresponding science. In a symbiotic relationship, technology has been stimulated by scientific interests and aided by scientific knowledge, and much scientific discovery has occurred in or been motivated by technology.

Science begins with *curiosity*, but technology starts with more mundane *needs*. A need is perceived and made more precise in what systems engineers call a set of *requirements*. Once that happens, any techniques available may be used to fill the prescription. For complex requirements a good deal of ingenuity is required, so we have come to call the people who transform the prescriptions into technology engineers. The ingenuity and experience needed for engineering has not required a developed science, but engineering has always been improved by the ability to measure. Comparison, matching, and duplication are engineering uses of measurement, and important for meeting requirements. Their usefulness was known by the time the Great Pyramids of Egypt were constructed (probably long before).

Today, we tend to link science and engineering because engineering is frequently able to call upon science to *predict* the outcomes of engineering processes that may be breaking new ground (not just ones that require matching parts or duplicating previous artifacts). The ability to predict is a key aspect of the understanding that scientific theories provide, and is a clear transfer from the ability to measure. But the use of substantial amounts of scientific understanding to improve engineering is relatively new because the development of scientific understanding has been slower than necessity-driven technology, requiring a similar but different kind of creativity.

From the standpoint of computational systems or physical systems, or their combination in robotics, we use all sorts of measurements in the construction process, but also in evaluating. We measure the performance of artifacts for engineering purposes, either to test the performance limits of a single artifact, to test its conformance to requirements, or to compare multiple artifacts. If it is for conformance, it may be done by matching quantitative behavior to requirements. If the requirements are qualitative, the number of requirements met and/or the degree to which they are met is interesting. Measurement can determine the success or failure of a portion of a technology project or of the entire project (or device, if that is the outcome of the project). But success is rarely absolute, and requirements met lead to ideas for better or stricter requirements. As Lord Kelvin pointed out, measurements provide a way of meeting these new requirements and thus of improving the product.

If project requirements are not easily translated into a behavioral outcome, then models of various suggested approaches can be developed and their behaviors may

stimulate the development process by people who tacitly know the needs but may not have been able to articulate a satisfactory set of requirements. The point is, however, that thinking about tests and measurements that might indicate the success or provide data for comparison can both prove and improve the outcome. This may be seen as the beginning of science, since scientific theories are models that meet certain requirements beyond those of a particular project.

Engineering is the process of creating artifacts, and "engineering sciences" developed by studying the process, are themselves sciences of the artificial. But for most engineering sciences, there are also underlying physical sciences. Because of that and because physical science provides the leading paradigm of science as it has developed over the ages, it is useful to consider examples of physical theoretical constructs. (It is well to keep in mind that informational theoretical constructs are going to be primary in computer science and artificial intelligence and a major consideration in robotics.)

Consider the construct *gravity*, which has a theoretical basis traceable to Galileo and Newton, refined more recently by Einstein. The gravitational constant is a part of that theory that has major technological ramifications, such as great predictive value in ballistic calculations. If one is building a catapult to bring down the walls of a fortified city, it could greatly help in making the right design decisions before actually building one; though such catapults were engineered well before Newton, or even Galileo.

Similarly, Newton's laws of motion are very useful, and *mass* is a fundamental theoretical construct used in both gravity and motion. When we create theoretical constructs like gravity and mass and can measure them, they increase our understanding of the physical world and our ability to predict how artifacts will perform in all situations: "To measure is to know."

Measurable theoretical constructs from physical theory influence design decisions and increase the likelihood of meeting technology prescriptions, with efficiencies of time, resources, and effort. The same thing will be true for theoretical constructs in information sciences and their related engineering branches as they develop.

In summary, science and technology both require measurements, and each has its separate needs. For technology, measurement is used to guide the engineering process and to check both the process and its products against requirements (the term often used is "validation and verification"). Often, however, intermediate measures can be found during the engineering process that turn out to have predictive value as to the final performance of an artifact. These measures may be indicative of important theoretical constructs that can enrich understanding within an underlying science. Science can exist by itself, and it originates in a basic human need to understand the world. Technology has existed for longer, as long as people have had a need for artifacts. Science and technology enrich each other, and measurement enriches them both.