Information Systems Engineering

David P. Silberberg and Glenn E. Mitzel

This article describes the science and technology vision of information systems engineering at APL. Information systems transform signal and data representations to high-level abstractions that enable people to perceive and interact with their environments. As information systems become more complex, they are expected to be more flexible, reusable, distributed, and extensible. To achieve these goals, information systems must be constructed upon solid software and system architecture foundations, and also must be created using sound software, cognitive, and information assurance methodologies. Since the dimensions of well-engineered information systems are too numerous to describe here, we cannot adequately cover all of their aspects. Therefore, we discuss key paradigms that will guide the future development of information systems at APL.

WHAT IS INFORMATION SYSTEMS ENGINEERING?

Information systems are computer-based infrastructures, organizations, personnel, and components that collect, process, store, transmit, display, disseminate, and act on information.[1] Information systems generally provide computer-based assistance to people engaging their environment as illustrated in Fig. 1, where engagements and environments are often too complex and dynamic to be handled manually.

Complex, dynamic engagements and environments require people to analyze and draw conclusions from an abstracted representation of the world, which enables them to make discrete decisions to achieve a desired effect in the world commensurate with their roles, tasks, and capabilities. The abstraction is sometimes portrayed as a hierarchy (Fig. 2) known as the Data, Information, Knowledge, and Wisdom (DIKW) paradigm.[2,3] The definitions of the layers are not precise, but the layers give a sense that data are processed to higher levels of abstraction, enabling people to make judgments about a situation and to follow a course of action. The widths of the elements of the hierarchy represent the relative volumes of data stored at each level. *Data* are the smallest symbolic units that describe measured or estimated phenomena. For example, sensors produce large volumes of data, but very little is understood by humans. *Information* here is used in a more restrictive sense than when we refer to information systems. In the DIKW paradigm, information is a more abstract understanding of the data derived by fusing data; it is typically the lowest layer where the symbolic units are interpretable by humans. *Knowledge* is belief about the information. In this layer, symbols are sufficiently abstract to enable

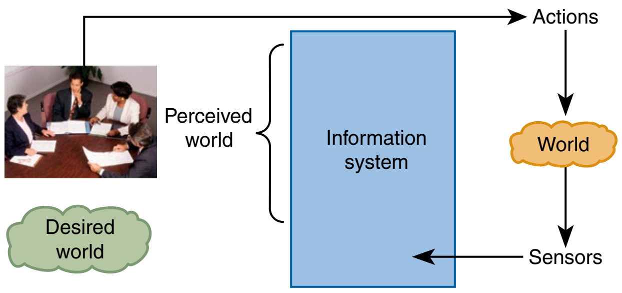

Figure 1. Information systems context.

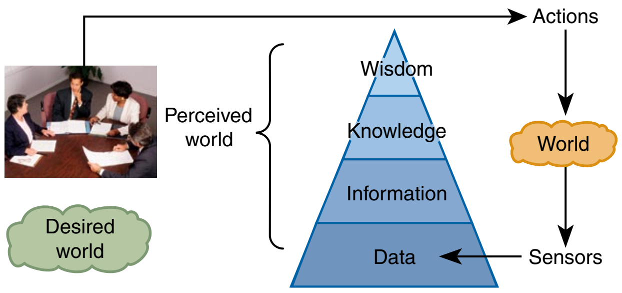

Figure 2. Information systems using the DIKW paradigm.

people to make decisions about and interact with their environments. Knowledge implies the combination of information with ancillary data and discernment of historical patterns. *Wisdom* is knowledge combined with insights and common sense. It is typically achieved by humans based on knowledge, information, and data. Unlike other levels of the DIKW hierarchy, wisdom is hard to derive automatically from lower-level information representations.

The *engineering* of information systems is the application of formal methods of analysis to create operational systems. Information systems that incorporate multiple technologies and processes must be designed and developed according to rigorous engineering standards to ensure that they support the requirements of their respective application domains and that they operate rapidly, accurately, and efficiently. Today, an exponentially increasing amount of data is available for processing and analysis, the number of decisions that must be made based on analysis is growing, the number of analysts that can make these decisions is decreasing, the time frame to make the decisions is becoming smaller, the size of information systems that support decision systems is increasing, and information systems are becoming more vulnerable to attack.

Military information systems must deal with denial and deceit, deliberate enemy actions to keep data from being collected or to distort the user's perception of the world to the point at which the user takes actions advantageous to the enemy. As the number of people needing to interact with information increases, the complexity of the corresponding information systems also increases. Data and information must be fused and synthesized so that fewer users can interact with a smaller amount of data at higher levels of abstraction. Information systems must present clear and reliable representations of their environments.

At APL, we are advancing the state of the art in the technologies and processes that will address these challenges. Intelligent use of data, information, and knowledge will enable systems to access and process large amounts of information more easily and rapidly. Automated decision systems will allow more rapid fusion of data from heterogeneous sources as well as apply polynomial-time approximations to exponential decision problems. Service- and agent-based technologies will enable developers to create increasingly complex systems by rapidly discovering and incorporating the capabilities of existing systems. Agile software engineering methodologies will promote the creation of more robust and flexible systems. Cognitive engineering systems will apply principles of analyzing user tasks and roles to streamlining user processes and to focusing users on the right level of abstraction for their task. Finally, information assurance (IA) methods will increase the integrity of information systems and reduce their vulnerabilities.

This article describes the science and technology (S&T) vision of selected technologies and processes at APL that allow our information systems to address these critical challenges.

APL'S S&T VISION FOR INFORMATION SYSTEMS ENGINEERING

Intelligent Use of Data, Information, and Knowledge

As the volume and heterogeneity of data play an increasingly prominent role in information systems, and as they continue to proliferate among many disparate organizations, advanced intelligent techniques must be exploited to simplify access to the data, accelerate data integration, and extract higher-level meaning from their content. Extracting higher-level meaning enables software to derive higher-level abstractions represented in the DIKW hierarchy. The APL S&T vision is to develop and use those techniques that will make access to data sources faster, easier, and less costly.

Data sources and their respective data management systems store and maintain information relevant to information systems. Data source representations generally fall into three categories: (1) structured (e.g., relational databases), (2) semi-structured (e.g., XML documents), and (3) unstructured (e.g., text documents). We envision that information systems incorporating new or legacy data sources of all three categories will use tools that provide simplified access to them and enable them to be integrated with other data sources and applications.

Applications and users requiring access to data sources often need detailed knowledge of the data source models, as well as the meanings and intent of the data terms, to formulate reasonable queries. A greater amount of knowledge is required to formulate queries to and integrate the results of heterogeneous data sources. Conceptual models and ontologies, which capture the structure and semantics of data sources, will play an increasingly important role in automated software that helps users gain access to these sources. Conceptual data models are machine-readable representations that describe the design of data sources, including the data represented, data groupings, and inter-data relationships. Ontologies are machine-readable representations that describe the semantics of the data sources, including the intended meaning of terms and relationships as well as their relationships to concepts outside the realm of what is represented in the data sources. Ontologies may also play a key role in integrating unstructured and structured data sources.

Using reasoning techniques over the conceptual models and ontologies, software tools will automatically formulate queries to one or more data sources, facilitate the process of integrating heterogeneous data sources, and enable reasoning about the data to provide higher-level abstractions that can be inferred from the data. The next section describes how decision models will fuse information provided by automated query and integration software to produce knowledge.

Figure 3 depicts this DIKW information processing hierarchy with respect to knowledge representation models, including conceptual models, ontologies, and decision models. Through the use of knowledge representation–supported capabilities, automated (or semi-automated) tools transform data to information to knowledge. Automated query tools that exploit conceptual schemas will enable simplified aggregation of information from individual data sources. Semi-autonomous data source integration tools will use ontologies to integrate information from heterogeneous data sources. Decision systems supported by decision models will enable information to be abstracted into knowledge. Applications that are supplied knowledge input from decision systems will further refine that knowledge. Finally, users interacting with applications that are supported by knowledge representation–based

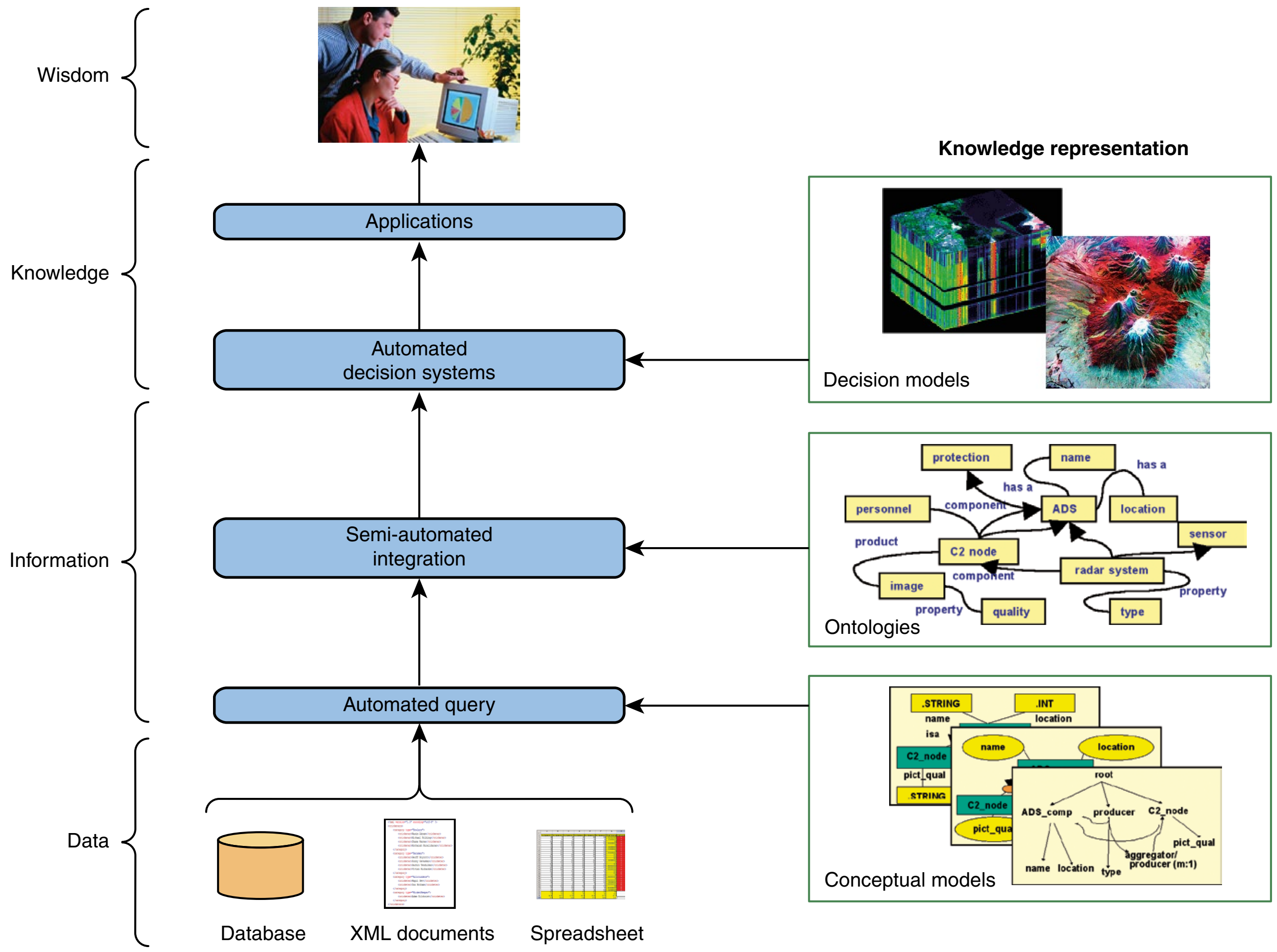

Figure 3. Information systems DIKW hierarchy using knowledge representation.

tools will gain wisdom to understand and act upon their environment at the abstraction level appropriate to their tasks.

Automated Decision Systems

Model-based Data Fusion

APL programs will increasingly rely on model-based data fusion techniques, which use models and model relationships to help automate the integration of sensor data to create both information and knowledge in the DIKW hierarchy. While overlaying sensor data are important in certain circumstances, model-based data fusion goes beyond sensor overlays by fusing information from multiple sensors to provide a picture of targets that have been detected, identified, and located with associated confidence. Furthermore, model-based data fusion provides support for sensor management and management of attack assets.

Traditionally, targeting for surface targets has been imagery centric and tracking has been point-data centric. Model-based data fusion integrates these two traditions. Tracking methods usually assume that sensor data are first processed to perform signal detection. The output of signal detection is observed point data in the form of kinematic quantities such as position, range rate, and time difference of arrival. Furthermore, signal detection provides signal-related data such as attributes and features that support the association of newly observed data with targets. Using techniques that include hypothesis testing and parameter and state estimation methods from statistical decision theory, the point data and signal-related data are then fused over time to provide target detection, data association, ID, and location. Imagery, on the other hand, can be fused at the pixel or feature level without first being processed to produce point data. Tracking and imagery fusion can be integrated by taking the results of imagery processing in the form of point- and signal-related data as input to tracking.

Model-based data fusion allows the best use to be made of what is known in the form of prior knowledge. Newly observed data are integrated with prior knowledge about targets, signatures, and sensors to rapidly produce the best information and knowledge. Essentially, this provides a continuous IPB (intelligence preparation of the battlefield) that allows quick reaction to threats. The model-based methods draw explicitly on mathematical models; examples include model-based automatic target recognition and kinematic tracking methods. More generally, techniques such as template matching and neural nets are also based on modeling assumptions. The techniques that can make the most efficient use of prior knowledge will generally provide the best performance when new observations are made.

Learning and Reasoning Tools

Learning is a key element of many intelligent systems. It is a fundamental way to acquire and assimilate new information to increase our knowledge of the world. Without learning, even the most seemingly intelligent entity is doomed to repeat the same mistakes endlessly.[4] Learning and reasoning tools deal with unanticipated change in the environment and help to improve software responses and behavior over time. Learning recognizes that software engineers cannot possibly conceive of all possibilities and plan for all contingencies. Learning tools identify new information and knowledge represented in the DIKW hierarchy.

The APL vision is to increasingly incorporate learning techniques that will address issues of scalability, model selection, communication constraints when operating in a distributed environment, and effective incorporation of domain knowledge. Example learning technologies are large-margin kernel machine and Bayesian belief networks (BBN). Large-margin classifiers such as support vector machines are discriminative methods that generalize well in sparse, high-dimensional settings. BBNs are probabilistic graphical models that provide a unified framework to manage computational uncertainty consistently through the fusion of computer science and probability theory. In particular, these graphical models enable robust incorporation of domain knowledge in machine learning and automated reasoning, which is difficult to achieve with traditional techniques based on statistical analysis and signal processing.

Distributed Computing

Web Services

The demand for near-universal access to data and applications will continue to grow throughout the next decade. The APL vision is to meet this need for future generations of command and control (C2) systems by using a service-oriented architecture (SOA), and, more specifically, by using a distributed web services approach.[5] Although data transfer to and from legacy applications has been simplified over the past decade by using technologies such as Microsoft's Component Object Model (COM) and OMG's Common Object Request Broker Architecture (CORBA), a more extensible and scalable architecture is now available.

Future global C2 systems are likely to be based on web-service architectures like the Net-Centric Enterprise Services (NCES) approach used to develop the Global Information Grid (GIG; Fig. 4). This approach allows groups of users, called communities of interest, to assemble on-the-fly groupings of data sources, display surfaces, decision support tools, and other services to meet their particular needs without having to bear the cost of development each time. Communities could be pulled together for short periods of time (e.g., for a single

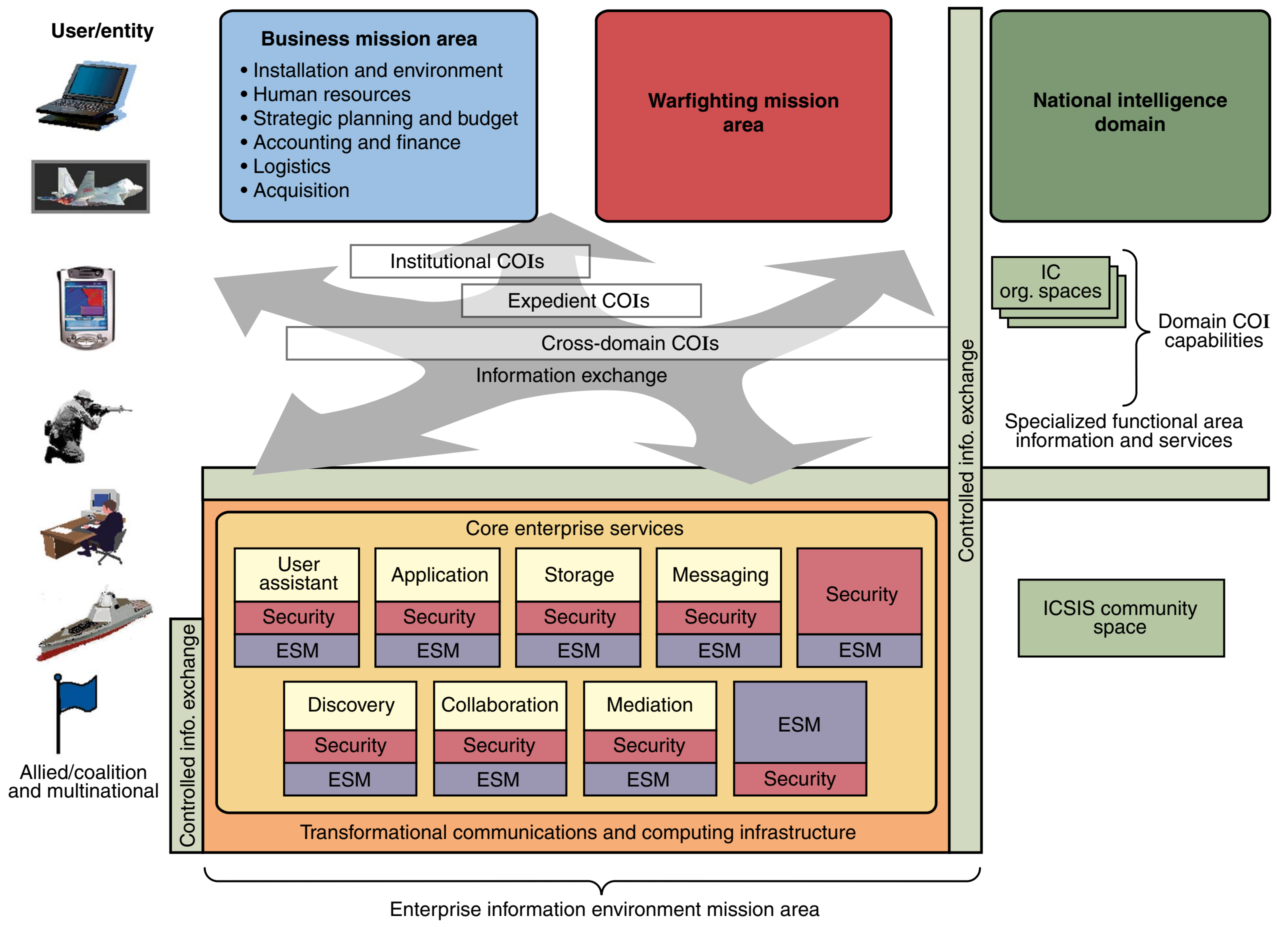

Figure 4. A holistic view of the GIG NCES. (Source: B. Appleby, NCES PM, Defense Information Systems Agency brief, "Net-Centric Enterprise Services OIPT," Apr 2004). (COI = communities of interest, ESM = enterprise service management, ICSIS = intelligence community system for information sharing.)

engagement) or for longer-standing timeframes (e.g., for years or even decades).

The core enterprise services are tied together using standard web services components, listed here with their responsibilities.

- *Discovery services* centralize services into a common registry and provide easy publish/find functionality. Currently handled via Universal Description, Discovery, and Integration (UDDI).
- *Description services* describe the public interface to a specific web service. Currently handled via the Web Service Description Language (WSDL).
- *Messaging services* encode messages in a common XML format so that they can be understood at either end. Currently includes XML Remote Procedure Call (XML-RPC) and Simple Object Access Protocol (SOAP).
- *Transport services* transport messages among applications using protocols such as HTTP, SMTP, or FTP.

Currently, web services are described in common UDDI registries by text that does not provide other applications with insight into the use and intent of the services. APL intends to develop ontology-based technologies that will enable applications to automatically discover and integrate services based on higher-level semantic descriptions.

Agent-based Systems

Agent-based systems are an emerging paradigm for constructing large, complex systems, and the APL S&T vision is to increase their incorporation into large information systems. Traditionally, large-scale systems are designed using the procedural approach achieved by functionally decomposing tasks into progressively smaller components until their coding can be managed by individual programming teams. Shortcomings of the procedural approach include the rigidity of the design and the fragility of the software. When requirements change, modifications may be needed to the software throughout the entire system. The object-oriented paradigm improves upon some of the problems of the procedural approach. The object-oriented approach requires systems to be broken into smaller components or objects that are abstractions of real world "things." Objects encapsulate state via variables and methods that operate on state. Objects also support inheritance. System modifications usually are made only to a few objects, leaving the rest of the system intact.

The agent paradigm extends the object-oriented paradigm in many important respects, enabling the creation of systems of multiple agents that are more flexible, adaptive, and self-organizing.[6] Agents are independent software components that exhibit autonomy, intelligence, the power to delegate, the ability to communicate, and sometimes mobility. *Autonomy* enables agents to act independently and with purpose. They solve goal-oriented requests of users and other agents but are not dependent on users and other agents for their operations. *Intelligence* enables agents to learn about their environments, to reason over knowledge they have acquired, and to make appropriate decisions. They may learn from their interactions with users to improve themselves and are adaptive to uncertainty and change. For example, agents may encapsulate automated decision systems and automated data access systems to exhibit. *Delegation* enables agents to call upon other agents to help solve problems, which does not preclude users from being "in the loop." *Communication* among agents is achieved through agent-communication languages, which are typically goal-oriented statements and requests. Since agents are autonomous and are created by authors from multiple domains, assistance from ontologies is required to translate communications from the language of one domain to another. *Mobility* enables agents to move among machines, gathering information from each platform to achieve its goals.

Sophisticated agent systems can allow agents to discover, communicate with each other, and self-organize to solve critical tasks. In a limited analogy, they can be compared to groups of people who organize to solve problems. The people are autonomous, have intelligence, and collaborate by using each other's expertise to achieve their goals.

Agent systems are anticipated to play a more prominent role in future APL development. These systems will enable large communities of software systems to form and exchange services and data more automatically. They will also help bridge the gaps among disparate, stove-piped systems to meet the needs of our sponsors.

Software Engineering

Much of the research in software engineering today, and for the foreseeable future, is rooted in one fundamental characteristic of software systems—increasing complexity. As our ability to build software and software systems improves, we build ever-larger and more complex systems. As complexity increases, a host of other issues arise. Systems must become more distributed because a single computer can no longer contain them. They also become more error-prone, and the errors become harder to find and fix. The teams needed to develop software systems also become larger, with the resultant increase in communication complexity and the requirement for more precision in their definition. Related to this problem is the one of defining the systems' behaviors in the first place, as they also become increasingly difficult for users to visualize and describe all aspects of those behaviors in advance. Finally, as the systems become ever larger and less deterministic, it becomes impossible to fully define and test all of their behaviors. Instead, systems must be developed that remain reliable and robust, even when handling conditions outside of design specifications.

In addressing the fundamental issue of system complexity, APL envisions applying aspects of three recent foci in software engineering research to internally developed systems. The first aspect is modeling to enable system developers to work with higher levels of abstraction, both in system specification and systems operations, using standards for describing modeling languages such as the Meta-Object Facility (MOF) as well as run-time behavior via the Model-Integrated Computing (MIC) effort. The second aspect of software engineering that APL will apply to its systems is new software development methodologies that are evolving as quickly as systems are. Keeping in mind that software development is not a "one-size-fits-all" prospect, APL will pursue agile technologies such as eXtreme Programming (XP), SCRUM, and the Agile Development Process (ADP) as well as more traditional approaches. The third aspect that APL will pursue is software architectures that exploit technologies such as the SOA and publish-subscribe infrastructures, and architecture frameworks such as J2EE, the DoD Architecture Framework (DoDAF), and Microsoft's .Net. These approaches will enable system developers to work at ever-higher levels of abstraction, managing the system development at the highest level—the architecture—as well as software. (See the article by Hanke et al., this issue.)

Cognitive Engineering

As the power of software and hardware systems increases and the amount of data with which the systems interact escalates, the requirement for more complex human interaction with greater volumes of data increases as well. Systems often must support multiple users with different needs for access to information and distinct system interaction roles. Unless the engineering of information systems considers user roles and tasks as a fundamental aspect of their engineering, users may suffer the consequence of system–user impedance mismatches. Ultimately, users need to interact with systems at a level of abstraction and ease to simplify their tasks as well as increase the understanding of the goals they are accomplishing. Thus, APL will increasingly incorporate *cognitive engineering* techniques in system development. The APL vision for cognitive engineering is discussed at greater length in the article by Gersh et al., this issue.

Information Assurance

The goal of IA is to ensure the confidentiality, integrity, and availability of information to authorized users and systems. *Confidentiality* is assurance that information is shared only among authorized people or organizations. *Integrity* is assurance that information is authentic and complete. It also means that the information can be trusted to be sufficiently accurate for its purpose. *Availability* is assurance that the systems responsible for delivering, storing, and processing information are accessible when needed by those who need them. Information systems deployed by APL will be IA enabled, which will raise the confidence level that our systems will operate reliably and consistently and will be more resistant to external threats. Even with a well-engineered information system, loss of one or more of these attributes can threaten the credibility of the information provided by the system. APL systems will increasingly integrate IA tools and methodologies in system development. Our IA vision is discussed at greater length in the article by Lee and Gregg, this issue.

SUMMARY

The engineering of information systems will increasingly rely on integrating a wide range of technologies and processes that enable users and systems to access, understand, and operate on large amounts of information. Information must be presented at the right level of abstraction at the right time to allow users to make informed and timely decisions. Thus, information system technologies will be engineered to integrate and process information across the DIKW continuum so that information is available at abstraction levels appropriate to user tasks and roles. APL will engineer state-of-the-art information systems by modeling data, information, and knowledge and by applying reasoning techniques to the models to automate information integration, fusion, and decision making. Furthermore, APL information systems will be created to participate in larger communities of interest by discovering other services and software agents and by making their services and agents available to other applications. APL software systems will be designed using the appropriate software engineering principles to ensure robustness and flexibility. In addition, good cognitive engineering techniques will be used to develop systems to ensure that users are working at appropriate levels of abstraction. Finally, IA tools and methodologies will be integrated into our information systems to protect their confidentiality, maintain their integrity, and ensure their availability.

ACKNOWLEDGMENTS: We would like to acknowledge Rosemary Daley, John Gersh, Peter Jacobus, Susan Lee, Jennifer McKneely, David W. Porter, Roger Remington, and I-Jeng Wang for their valuable inputs to this article.

REFERENCES

[1]*Doctrine for Command, Control, Communications, and Computers (C4) Systems Support to Joint Operations*, Joint Pub 6-0 (30 May 1995); http://www.dtic.mil/doctrine/jel/new_pubs/jp6_0.pdf.
[2]Ackoff, R. L., "From Data to Wisdom," *J. Appl. Syst. Anal.* **16**, 3–9 (1989).
[3]Cleveland, H., "Information as Resource," *The Futurist*, 34–39 (Dec 1982).
[4]Yoon, B., "Get Smart: Real World Learning"; http://www.darpa.mil/DARPAtech2004/pdf/scripts/YoonRWLScript.pdf (2004).
[5]Erl, T., *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Prentice Hall (2004).
[6]Hendler, J., "Agents and the Semantic Web," *IEEE Intel. Syst.* **16**(2), 30–37 (2001); http://www.cs.umd.edu/users/hendler/AgentWeb.html.

THE AUTHORS

David P. Silberberg is a member of APL's Principal Professional Staff and serves as the Assistant Supervisor of the System and Information Sciences Group of the Research and Technology Development Center. In addition, Dr. Silberberg is an Assistant Research Professor in the Department of Computer Science at JHU and teaches courses in both distributed database theory and XML technologies at the JHU Engineering and Applied Science Programs for Professionals. Previously, he was a principal architect of the Hubble Space Telescope Data Archive and Delivery Service and the NASA National Science Space Data Center Archive. Dr. Silberberg received both S.B. and S.M. degrees in computer science from MIT in 1981 and a Ph.D. in computer science from the University of Maryland, College Park, in 2002.

David P. Silberberg

Glenn E. Mitzel is a member of APL's Principal Professional Staff and serves as the Chief Scientist for the Precision Engagement Business Area. From 1991 to 2000, Dr. Mitzel supervised the Ship Systems Group in the Power Projection Systems Department. He has worked on a variety of new concepts and techniques in remote surveillance and targeting, including geospatial pattern recognition for ship discrimination, passive ranging of jamming aircraft, multimodal data fusion, and space-based surveillance. Dr. Mitzel received B.S.E., M.S.E., and Ph.D. degrees in electrical engineering from JHU in 1973, 1975, and 1978, respectively. For further information, contact Dr. Silberberg at david.silberberg@jhuapl.edu.

Glenn E. Mitzel