Foundations of Data Fusion for Automation

©EYEWIRE

Developing paradigms of data fusion for sensor-actuator networks that perform engineering tasks.

S.S. Iyengar, S. Sastry, and N. Balakrishnan

Data fusion is a paradigm for integrating data from multiple sources to synthesize new information such that the whole is greater than the sum of its parts. This is a critical task in contemporary and future systems that have distributed networks of low-cost, resource-constrained sensors [1], [2]. Current techniques for data fusion are based on general principles of distributed systems and rely on cohesive data representations to integrate multiple sources of data. Such methods do not extend easily to systems in which real-time data must be gathered periodically by cooperative sensors, where some decisions become more critical than other decisions episodically.

There has been extensive study in the areas of multisensor fusion and real-time sensor integration for time-critical sensor readings [3]. A distributed sensor data network is a set of spatially scattered sensors designed to derive appropriate inferences from the information gathered. The development of such networks for information gathering in unstructured environments is receiving much interest, partly because of the availability of new sensor technology that is economically feasible to implement [4]. Sensor data networks represent a class of distributed systems that are used for sensing and in-situ processing of spatially and temporally dense data from limited resources and harsh environments, by routing and cooperatively processing the information gathered. In all these systems, the critical step is the fusion of data gathered by sensors to synthesize new information.

Our interest is in developing paradigms of data fusion for sensor-actuator networks that perform engineering tasks. We use automation systems as an illustrative example. Automation systems represent an important, highly engineered, trillion-dollar business in the United States.

1094-6969/03/$17.00©2003IEEE

The real-time and distributed nature of these systems, with the attendant demands for safety, determinism, and predictability, represent significant challenges and, hence, these systems are a good example. An automation system is a collection of devices, equipment, and networks that regulates operations in a variety of manufacturing, material- and people-moving, monitoring, and safety applications.

Automation systems evolved from early centralized systems to large distributed systems that are difficult to design, operate, and maintain [5]. Current hierarchical architectures, the nature and use of human-computer-interaction (HCI) devices, and the current methods for addressing and configuration increase system life-cycle costs. Current methods to integrate system-wide data are hardcoded into control programs and are not based on an integrating framework. Legacy architectures of existing automation systems are unable to support future trends in distributed automation systems [6]. Current methods for data fusion are also unlikely to extend to future systems because of system scale and simplicity of nodes.

We present a new integrating framework for data fusion that is based on two systems concepts: a conceptual framework and the goal-seeking paradigm [7]. The conceptual framework represents the structure of the system, and the goal-seeking paradigm represents the behavior of the system. Such a systematic approach to data fusion is essential for proper functioning of future sensor-actuator networks [8] and SmartSpace [9]. A SmartSpace is a new approach to HCI that has been ushered in by the emerging trends in pervasive computing. It is an intelligent, integrated environment that gathers automation systems data, facilitates multiple users to simultaneously interact with an automation system, and facilitates holistic human decision-making by integrating system-wide data. In the short term, such techniques help to infuse emerging paradigms into existing automation architectures. We must bring together knowledge in the fields of sensor fusion, data and query processing, automation systems design, and communication networks to develop the foundations. While extensive research is being conducted in these areas, we hope that researchers in related areas will venture into this emerging and important area of research [2].

Automation Systems

An automation system is a unique, distributed, real-time system that comprises a collection of sensors, actuators, controllers, communication networks, and user-interface devices. Such systems regulate the coordinated operation of physical machines and humans to perform periodic and precise tasks that may sometimes be dangerous for humans. Examples of automation systems are: a car-manufacturing factory, a baggage-handling system in an airport, and an amusement park ride. Part, plant, and process are three entities of interest.

- A part is a basic unit passing through the automation system that receives attention from the plant and processes.
- A plant comprises a collection of stations, mechanical fixtures, energy resources, and control equipment that regulates operations by using a combination of mechanical, pneumatic, hydraulic, electric, and electronic components or subsystems.

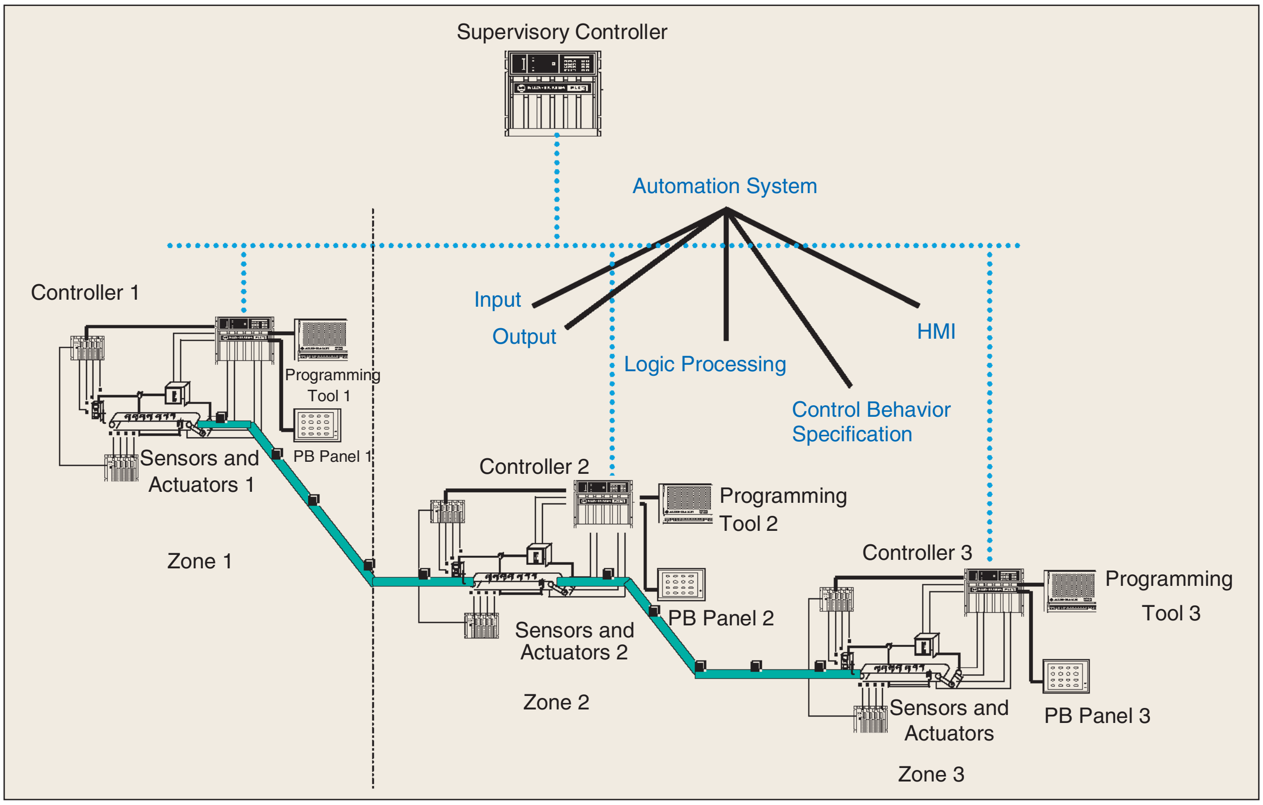

Fig. 1. Major aspects of an automation system.

- A process specifies a sequence of stations through which a part must traverse and operations that must be performed on the part at each station.

Figure 1 shows the major aspects of an automation system. All five aspects, namely input, output, logic processing, behavior specification, and HCI, must be designed, implemented, and commissioned to successfully operate an automation system. Sensors and actuators are transducers that are used to acquire inputs and set outputs, respectively. The controller periodically executes logic to determine new output values for actuators. HCI devices are used to specify logic and facilitate operator interaction at runtime.

The architecture of existing automation systems is hierarchical, and the communication infrastructure is based on proprietary technologies that do not scale well. Ethernet is emerging as the principal control and data exchange network. The transition from rigid, proprietary networks to flexible, open networks introduces new problems into the domain of automation systems, and security is a critical problem that demands attention.

System Characteristics

Automation systems operate in different modes. For example, "*k*-hour-run" is a mode that is used to exercise the system without affecting any parts. Other examples of modes are "automatic," "manual," and "semiautomatic." In all modes, the overriding concern is to achieve deterministic, reliable, and safe operations. The mode of the system dictates the degree to which humans interact with the system. Safety checks performed in each of these modes usually increase with the degree of user interaction.

Communication traffic changes with operating mode. For example, when the system is in automatic mode, limited amounts of data (a few bytes) are exchanged; such data flows occur in localized areas of the system. In this mode, small disruptions and delays in message delivery can be tolerated. When there is a disruption (in part, plant, or process), however, large bursts of data will be exchanged between a large number of nodes across the system. Under such conditions, disruptions and delays significantly impair system capabilities.

Because of both the large capital investments required and the changing market place, these systems are designed to be flexible with respect to part, plant, and process. Automation systems operate in a periodic manner. The lack of tools to evaluate performance makes it difficult to evaluate the system's performance and vulnerability. These systems are highly engineered and well documented in the design and implementation stages. Demands for reconfigurable architectures and new properties such as self-organization require a migration away from current hierarchical structures to loosely coupled networks of devices and subsystems. The dominant language that is currently used to specify control behavior is called Ladder and it is a part of the IEC 61131-3 specification. Most of the language editors that are used to specify such control behavior provide support for both online and offline programming, debugging, and monitoring activities.

Operational Problems

Hierarchical architecture and demands for backward compatibility create a plethora of addressing and configuration problems. Cumbersome and expensive implementations and high commissioning costs are a consequence of such configuration problems. Figure 2 shows a typical configuration of input and output (IO) points connected to controllers.

Unique addresses must be assigned to every IO point and single structural component in Figure 2. These addresses are typically related to the jumper settings on the device or chassis; thus, a large number of addresses are related and structured by manual naming conventions. In addition, depending on the settings on device or chassis, several items in software packages must also be configured manually. Naturally, such a landscape of addresses leads to configuration problems. State-of-the-art implementations permit specification of a global set of tags and maintain multiple links behind the scenes. While this simplifies the user's chores, the underlying problems remain. The current methods of addressing and configuration will not extend to future systems that are characterized by large-scale, complex interaction patterns and emergent behaviors.

Current methods for commissioning and fault recovery are guided by experience and based on trial and error. User-interface devices display localized, controller-centric data and do not support holistic, system-wide decision making. Predictive nondeterministic models do not accurately represent system dynamics and, hence, approaches based on such models have met with limited success. The current practice is a template-based approach that is encoded into Ladder programs. These templates recognize a few commonly occurring errors.

Despite these operational problems, automation systems are a benchmark for safe, predictable, and maintainable systems. HCI devices are robust and reliable. Methods for safe interaction, such as the use of dual palm switches, operator level debouncing, and safety mats, are important elements of any safe system. Mechanisms that are used to monitor, track, and study trends in automation systems are good models for such tasks in general, distributed systems.

Benefits of Data Fusion

Data fusion can alleviate current operational problems and support the development of new architectures that preserve the system's characteristics. For example, data fusion techniques based on the foundations discussed in this article can provide a more systematic approach to commissioning, fault management (detection, isolation, reporting, and recovery), programming, and security management. Data fusion techniques can be embodied in distributed services that are appropriately located in the system and such services support a SmartSpace in decision making [9].

Foundations of Data Fusion

The principle issue for data fusion is to manage uncertainty. We accomplish this task through a goal-seeking paradigm. A conceptual framework simplifies application of the goal-seeking paradigm in the context of a multilevel system, such as an automation system.

Representing System Structure

A conceptual framework is an experience-based stratification of the system, as shown in Figure 3. We see a natural organization of the system into levels of a node, station, line, and plant. At each level, the dominant considerations are depicted on the right. There are certain cross-cutting abstractions that do not fit well into a hierarchical organization. The neighborhood of a node admits accessible nodes that are not necessarily in the same station. For example, there may be a communications node that has low load that could perform data fusion tasks for a duration when another node in the neighborhood is managing a disruptive fault. Similarly, energy resources in the environment affect all levels of the system.

A conceptual framework goes beyond a simple layering approach or a hierarchical organization. The strata of a conceptual framework are not necessarily organized in a hierarchy. The strata do not provide a service abstraction to other strata like a layer in a software system. Instead, each stratum imposes performance requirements for other strata to which it relates. At runtime, each stratum is responsible for monitoring its own performance based on the sensed data. The system will perform as expected as long as the monitored performance is within tolerance limits specified in the goal-seeking paradigm.

Representing System Behavior

We represent the behavior of the system using the goal-seeking paradigm. We briefly review the fundamental state transition paradigm and certain problems associated with this paradigm before discussing the goal-seeking paradigm.

STATE TRANSITION PARADIGM

The state-transition paradigm is an approach to modeling and describing systems based on two key assumptions: first, that states of a system are precisely describable; and second, that the dynamics of the system are also fully describable in terms of states, transitions, inputs that initiate transitions, and outputs that are produced in states.

A state-transition function $S_1 : Z_{t1} \otimes X_{t1,t2} \rightarrow Z_{t2}$ defines behavior of the system by mapping inputs to a new state. For each state, an output function $S_2 : Z_{ti} \rightarrow \psi_{ti}$, where $X_{t1,t2}$ is the set of inputs presented to the system in the time interval between $t1$ and $t2$; ψ_i is the set of outputs produced at time ti; Z_{t1} and Z_{t2} are states of the automation system at times $t1$ and $t2$, respectively; and $\otimes$, the Cartesian product, indicates that the variables before the arrow (i.e., inputs $X_{t1,t2}$ and Z_{t1}) are causes for change in the variables after the arrow (i.e., outputs Z_{t2}).

To understand such a system, one needs to have complete data on Z_{t1} and $X_{t1,t2}$ and knowledge about S_1 and S_2. This paradigm assumes that only a lack of data and knowledge prevents us from completely predicting the future behavior of a system. There is no room for uncertainty or indeterminism. Such a paradigm (sometimes called the input–output or stimulus-response paradigm) can be useful in certain limited circum-

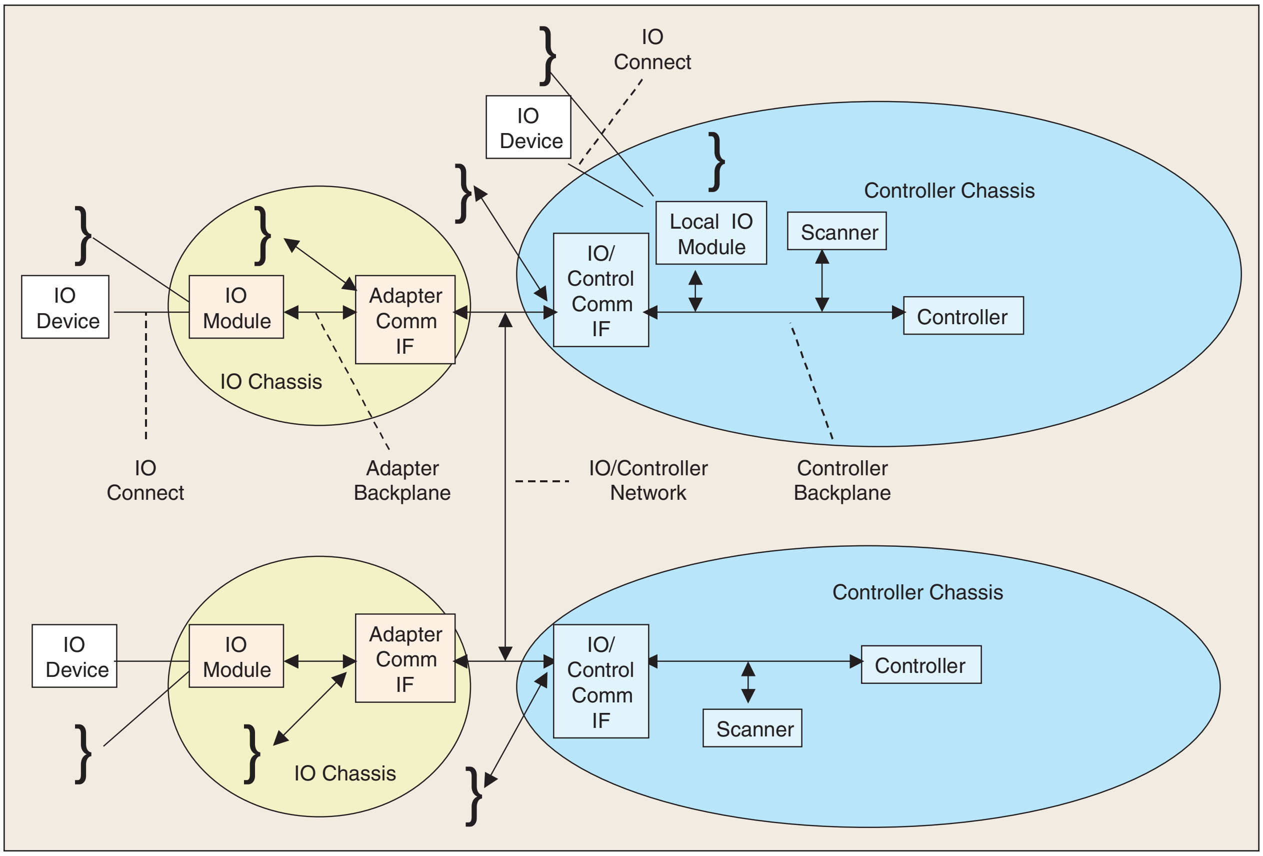

Fig. 2. Connecting input–output points to controllers.

stances for representing the interaction between two systems; it can be erroneous if it is overextended. There is no consistent or uniform definition of a state, and contextual information that is based on collections of states is ignored. In such circumstances, the specifications in a state-transition paradigm are limited by what is expressed and depends heavily on environmental influences that are received as inputs. While it appears that the state-transition paradigm is simple, natural, and easy to describe, such a formulation can be misleading, especially if the true nature of the system is goal seeking.

GOAL-SEEKING PARADIGM

A goal-seeking paradigm is an approach to modeling and describing systems that explicitly supports uncertainty management by using additional artifacts and transformations. The system can choose actions from a range of alternate actions, Π, in response to events occurring or expected to occur. These actions represent a choice of decisions that can be made in response to a given or emerging situation.

There is a range of uncertainties, Δ, that impact the success of selected decisions. Uncertainties arise from two sources: first, from an inability to correctly anticipate inputs, either from the automation system or from users; and second, from an incomplete or inaccurate view of the outcome of a decision, even if the input is correctly anticipated. For example, an operator may either switch the mode of the system from "automatic" to "manual" by mistake, malicious intent, or poor training. Assuming that the user made appropriate choices, the outcome of a decision can still be uncertain because a component or subsystem of the automation system may fail just before executing the decision.

The range of consequences, Ψ, represent outcomes that result from an implementation of system decisions. Consequences are usually outputs that are produced by the system. Some of these outputs may be consumed by users to further resolve uncertainties, and other outputs may actuate devices in the automation system.

The system maintains a reflection, $\Xi : \Pi \times \Delta \to \Psi$, which is its view of the environment. Suppose that the system makes a decision, $\pi \in \Pi$; the system benefits from an understanding of what consequence $\pi \in \Pi$, ψ produces. The consequence ψ is presented as an output either to humans within a SmartSpace or to the automation system. ψ does not obviously follow ψ as specified by S_2 because of uncertainties in the system.

The evaluation set, Λ, represents a performance scale that is used to compare the results of alternate actions. That is, suppose the system could make two decisions, $\pi_1 \in \Pi$ or $\pi_2 \in \Pi$, and these decisions have consequences $\psi_1, \psi_2 \in \Psi$, respectively. Λ helps to determine which of the two decisions are preferable.

An evaluation mapping, $\Omega : \Psi \times \Pi \to \Lambda$, is used to compare outcomes of decisions using the performance scale. Λ is specified by taking into account the *extent or cost of the* effort associated with a decision; i.e., $\pi \in \Pi$. For any $\pi \in \Pi$ and $\pi \in \Pi$, Ω assigns a value $\lambda \in \Lambda$ and helps to determine the system's preference for a decision-consequence pair (π, ψ).

A tolerance function, $\Gamma : \Pi \times \Delta \to \Delta$, indicates the degree of satisfaction with the outcome if a given uncertainty $\delta \in \Delta$ comes to pass. For example, if the conditions are full of certainty, the best or optimal decision can be identified. If, however, there are several events that are anticipated (i.e., $|\Delta| \gg 1$), the performance of the system, as evaluated by Ω, can be allowed to deteriorate for some $\delta \in \Delta$ but this performance must stay within a tolerance limit that will ensure survival of the system.

Based on the above artifacts and transformations, the functioning of a system, in a goal-seeking paradigm, is defined as: Find a decision $\pi \in \Pi$ so that the outcome is acceptable (e.g., within tolerance limits) for any possible occurrence of uncertainty $\delta \in \Delta$; i.e., $\Omega(\Xi(\pi, \delta), \pi) \leq \Gamma(\delta, \pi), \forall\delta \in \Delta$

Security Management for Discrete Automation

Security management in a discrete automation system is critical because of the current trends toward using open communication infrastructures. Automation systems present challenges of traditional distributed systems, emerging sensor networks, and the need to protect the high investments in contemporary assembly lines and factories. Formulating the security management task in the state-transition paradigm is a formidable task and perhaps may never be accomplished because of the scale and the uncertainties that are present. We demonstrate how the goal-seeking formulation helps and the specific data fusion tasks that facilitate security management.

A future automation system is likely to comprise large sensor-actuator networks as important subsystems [8]. Nodes in such a system are resource constrained. Determinism and low jitter are extremely important design considerations. Since a node typically contains a simple processor without any controller hierarchy or real-time operating system, it is infeasible to implement a fully secure communications channel for all links without compromising determinism and performance. A distinguishing feature of the new sensor networks is that they do not have enough resources to run a large RTOS; if they did, then each node would become much more expensive, and that

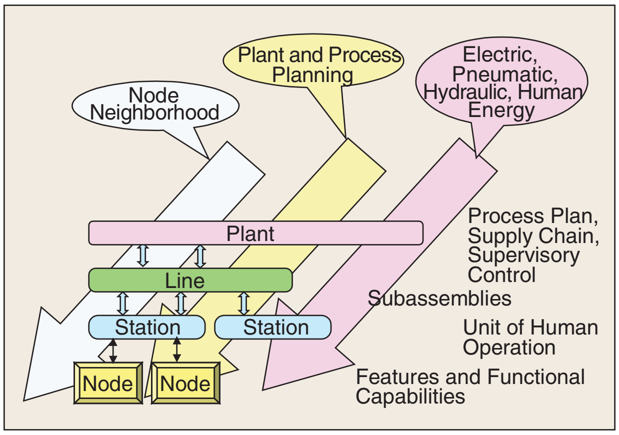
Node Neighborhood
Plant and Process Planning
Electric, Pneumatic, Hydraulic, Human Energy
Plant
Line
Station
Station
Node
Node
Process Plan, Supply Chain, Supervisory Control
Subassemblies
Unit of Human Operation
Features and Functional Capabilities

Fig. 3. Conceptual framework for an automation system.

eats into the scale of the system you can build. You can easily put together a million-node system if the nodes are very inexpensive but only 100 nodes if each node costs 10,000 times more. For the same reason, there is no controller hierarchy, i.e., a special processor that performs signal processing, memory management, or communications tasks. The best that you can do is perhaps a 4-MHz processor with a microkernel OS that is good for executing a single thread task. If this processor is communicating, it cannot be sensing or actuating, and if it is doing physical IO, it cannot be communicating. Further, because of the large installed base of such systems, security mechanisms must be designed to mask the uneven conditioning of the environment [10]. The goal-seeking formulation, presented here, and the cooperative data fusion tasks associated support such an implementation.

Goal-Seeking Formulation

This section details the artifacts and transformations necessary for security management.

ALTERNATE ACTIONS

The set of actions includes options for security mechanisms that are available to a future automation system. Asymmetric cryptography may be applied to a specific link, a broadcast, a connection, or a session. Digital signatures or certificates may be required. Frequency hopping may be used to make it difficult for malicious intruders to masquerade messages or eavesdrop. Block ciphering, digest function, or μ-T might be applied. These possible actions comprise the set of alternate actions. At each time step, the automation system selects one or more of these alternate actions to maintain security of the system. Because the nodes are resource constrained, it is not possible to implement full encryption for each of the links. Thus, to make the system secure, one or more mechanisms must be selected in a systematic manner depending on the current conditions in the system.

UNCERTAINTIES

As already discussed, there are two sources of uncertainties. First, the system cannot guess the choice when a specific user choice is expected as input (a subjective decision). Second, malfunctions of a component or subsystem cannot be predicted. For example, an established channel, connection, packet, or session may be lost. A node receiving a request may be unable to respond to a query without compromising local determinism or jitter. The node, channel, or subsystem may be under a denial of service attack. There may be a malicious eavesdropper listening to the messages, or someone may be masquerading production data to mislead management.

CONSEQUENCES

It is not possible to predict either the occurrence or the time of occurrence of an uncertainty if it occurs. The selected actions may not lead to intended consequences if one or more uncertainties come to pass, however. Some example consequences are when the communications channel is highly secure at the expected speeds, the channel may be secure at a lower speed, or the channel may be able to deliver data at desired speed without any security. The supplied authentication, digital signature, or certificate is either verified or not verified.

REFLECTION

This transformation maps every decision-uncertainty pair into a consequence that is presented as an output. The reflection includes all possible choices, without any judgment about either the cost of effort or feasibility of the consequence in the current system environment.

EVALUATION SET

The three parameters of interest are security of channel, freshness of data, and authenticity of data. Assuming a binary range for each of these parameters, we get the following scale to evaluate consequences:

1) highly secure channel with strongly fresh, truly authenticated data
2) highly secure channel with strongly fresh, weakly authenticated data
3) highly secure channel with weakly fresh, truly authenticated data
4) highly secure channel with weakly fresh, weakly authenticated data
5) weakly secure channel with strongly fresh, truly authenticated data
6) weakly secure channel with strongly fresh, weakly authenticated data
7) weakly secure channel with weakly fresh, truly authenticated data
8) weakly secure channel with weakly fresh, weakly authenticated data
9) total communication failure, no data sent.

EVALUATION MAPPING

Ideally, consequences follow from decisions directly. Because of uncertainties, the consequence obtained may be more desirable or less desirable, depending on the circumstances. Evaluation mapping produces such an assessment for every consequence-decision pair.

TOLERANCE FUNCTION

The tolerance function establishes a minimum performance level on the evaluation set that can be used to decide whether or not a decision is acceptable. In an automation system, the tolerance limit typically changes when the system conditions are different. For example, a highly secure channel with strongly fresh, truly authenticated data may be desirable when a user is trying to reprogram a controller and weakly secure channel with weakly fresh, weakly authenticated data may be adequate during commissioning phases. The tolerance function can be defined to include such considerations as operating mode and system conditions with the view of evaluating a decision in the current context of a system.

Applying Data Fusion

Only a few uncertainties may come to pass when a particular alternate action is selected by the system. Hence, we first build a list of uncertainties that apply to every alternate action $\{\Delta_1, \Delta_2, \ldots \Delta_{|\Pi|}\}$. For an action, π_k, if the size of the corresponding set of uncertainties are $|\Delta_k| > 1$, then data fusion must be applied.

To manage security in an automation system, we need to work with two types of sensors. The first type is sensors that are used to support the automation system, such as ones that sense presence of a part, completion of a traversal, or failure of an operation. In addition, there are sensors that help resolve uncertainties. Some of these redundant sensors could either be additional sensors that sense a different modality or another sensor whose value is used in another context to synthesize new information. Some uncertainties can only be inferred as the absence of certain values. For every uncertainty, the set of sensor values, or lack thereof, and inferences are recorded a priori. Sensors that must be queried for fresh data are also recorded a priori.

For each uncertainty, we also record the strata in the conceptual framework that dominates the value of an uncertainty. For example, suppose there is an uncertainty in the mode of a station. The resolution of this uncertainty is based on whether or not there is a fault at the station. Suppose there is a fault in the station; then the mode at the station must dominate the mode at the line to which the station belongs. Similarly, when the fault has been cleared, the mode of the line must dominate if the necessary safety conditions are met. Thus, the conceptual framework is useful for resolving uncertainties. The specific implementation of data fusion mechanisms will depend on the capabilities of the nodes. In a resource-constrained environment, we expect such techniques to be integrated with the communications protocols.

Summary

In this article we provided a new foundation for data fusion based on two concepts: a conceptual framework and the goal-seeking paradigm. The conceptual framework emphasizes the dominant structures in the system. The goal-seeking paradigm is a mechanism for representing system evolution that explicitly manages uncertainty. The goal-seeking formulation for data fusion helps to distinguish between subjective decisions that resolve uncertainty by involving humans and objective decisions that can be executed by computers. These notions are useful for critical tasks such as security management in large-scale distributed systems. Investigations in this area, and further refinement of the goal-seeking formulation for instrumentation and measurement applications, are likely to lead to future systems that facilitate holistic user decision making.

Acknowledgments

This work is supported in part by the University of Akron, College of Engineering Research Startup Grant, 2002-2004 to S. Sastry, and an NSF Grant IIS-0239914 to S.S. Iyengar.

References

[1] R.R. Brooks and S.S. Iyengar, *Multi-Sensor Fusion: Fundamentals and Applications with Software*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[2] S.S. Iyengar, K. Chakrabarty, and H. Qi, "Distributed sensor networks for real-time systems with adaptive configuration," *J. Franklin Inst*., vol. 338, pp. 571–582, 2001.

[3] R. Kannan, S. Sarangi, S.S. Iyengar, and L. Ray, "Sensor-centric quality of routing in sensor networks," in *Proc. IEEE Infocom*, Apr. 2003, pp. 692–701.

[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Netw*., vol. 38, pp. 393–422, 2002.

[5] J. Agre, L. Clare, and S. Sastry, "A taxonomy for distributed real-time control systems," *Adv. Comput*., vol. 49, pp. 303–352, 1999.

[6] D. Slansky, "Collaborative discrete automation systems define the factory of the future," *ARC Strategy Report*, May 2003.

[7] M.D. Mesarovic and Y. Takahara, *Mathematical Theory of General Systems*. New York: Academic, 1974.

[8] S. Sastry, "Sensor-actuator networks for automation," Dept. Elect. Comput. Eng., Univ. Akron, Akron, OH, Tech. Rep. ECE-PAL-2003-16, 2003.

[8] S. Sastry, "Foundations of a SmartSpace for automation," in *Proc. Int. Conf. Artificial Intelligence*, Las Vegas, NV, June 23-26, 2003, pp. 3–9.

[9] M. Sathyanarayanan, "Pervasive computing: Vision and challenges," *Pers. Comput*., vol. 8, no. 4, pp. 10–17, Aug. 2001.

S.S. Iyengar is a Distinguished Professor of Computer Science and is currently the Roy Paul Daniels Professor of Computer Science and chair of the Department of Computer Science in Louisiana State University. He has authored or co-authored several books and over 300 research papers in various areas of computer science and is one of the pioneers in the area of distributed sensor networks. His research is funded by several federal agencies including NSF and DARPA. He is a Fellow of the IEEE and ACM.

S. Sastry is an assistant professor in the Department of Electrical and Computer Engineering at the University of Akron. He has over a dozen years of industry experience. He has published several research papers and holds two patents related to automation. His research interests are large-scale distributed systems and sensor actuator networks.

N. Balakrishnan is a divisional chair at the Indian Institute of Science and is a professor at the Supercomputer Education and Research Center. He has published a large number of papers in computational science and is the recipient of the PADMASHRI award from the Government of India. His areas of research include computational electromagnetics, multiparametric radars, signal coding, and distributed sensor networks and information technology. He is a founding director for several IT initiatives based at the Indian Institute of Science.