# Information fusion for process control

**Ľubomír Petrovič, Ladislav Jurišica**

**Abstract**

The paper presents issues of information integration and information fusion for process control. At first primary requirements for sensor system and categories for sensor types are described with analysis of sensor configuration types. New approaches for system design with information integration in different time models are described. In conclusion we describe architectures and models for information integration.

**Key words:** information integration, sensor fusion, real-time systems, time triggered systems, event triggered systems, information integration architectures and models.

## Introduction

Present applications, such as industrial, medical, military, application for safety and transportation applications depend on embedded computer system that interact with the real world. Especially dependable reactive systems that have to provide a critical real-time service need carefully design and implementation. There is need to consider four basic requirements:

**Requirements for sensors:** Due to limited resolution, cross-sensitivity, measurement noise, and possible sensory deprivation, an application may never depend on particular sensor information.

**Real-time requirements:** In many cases, operations have to be carried out with respect to real time. Timing failures in such applications may endanger operator and machine.

**Dependability requirements:** Since embedded systems are often integrated into larger systems that depend on embedded subsystems, the embedded systems have to be designed and implemented in a way that they provide a robust service. An embedded system might have to provide a particular service even in case of failure of some of its components. Such fault-tolerant behavior requires a proper design of a system with regard to the possible failure modes of its components.

**Complexity management requirements:** There is often need to split a complex system, such as the software of a robot with distributed sensors and actuators, into small comprehensible subsystems in order to ease implementation and testing.

## Information integration – current state

There is confusion in the terminology for fusion systems. The terms "sensor fusion", "data fusion", "information fusion", "multi-sensor data fusion", and "multi-sensor integration" have been widely used in the technical literature to refer to a variety of techniques, technologies, systems, and applications that use data derived from multiple information sources.

Fusion applications range from real-time sensor fusion for the navigation of mobile robots to the off-line fusion of human or technical strategic intelligence data.

Several attempts have been made to define and categorize fusion terms and techniques. For example: "data fusion" can be used as the overall term for fusion. However, while the concept of data fusion is easy to understand, its exact meaning varies from one scientist to another. A literal definition of information fusion can be found at the homepage of the International Society of Information Fusion [ISIF]. ISIF is a organization that associates scientists oriented to area of information integration. This organization defines information fusion as:

***Information Fusion*** *encompasses theory, techniques and tools conceived and employed for exploiting the synergy in the information acquired from multiple sources (sensor, databases, information gathered by human, etc.) such that the resulting decision or action is in some sense better (qualitatively or quantitatively, in terms of accuracy, robustness, etc.) than would be possible if any of these sources were used individually without such synergy exploitation.*

By defining a subset of information fusion, the term sensor fusion is introduced as:

***Sensor Fusion*** *is the combining of sensory data or data derived from sensory data such that the resulting information is in some sense better than would be possible when these sources were used individually.*

## Categorization of sensor configurations types

In general sensor systems can be created from many different types of sensors which information and data format are also different. There is need to combine these data formats into one readable format. Applications have to be designed in the way of independency, because each sensor can be depended on another sensor. If one of system sensors crash other sensors have to fully or partly represent work of broken sensor.

Sensor fusion networks can also be categorized according to the type of sensor configuration. Durrant-Whyte [Durrant88] distinguishes three types of sensor configuration:

**Complementary**: A sensor configuration is called complementary if the sensors do not directly depend on each other, but can be combined in order to give a more complete image of the phenomenon under observation. This resolves the incompleteness of sensor data. Sensor S2 and S3 in figure 1 represent a complementary configuration, since each sensor observes a different part of the environment space.

**Competitive**: Sensors are configured competitive if each sensor delivers independent measurements of the same property. There are two possible competitive configurations - the fusion of data from different sensors or the fusion of measurements from a single sensor taken at different instants. Competitive sensor configuration is also called a redundant configuration [LuoKay89]. A special case of competitive sensor fusion is fault tolerance. Fault tolerance requires an exact specification of the service and the failure modes of the system. In case of a fault covered by the fault hypothesis, the system still has to provide its specified service. Sensor S1 and S2 in figure 1 represent a competitive configuration, where both sensors redundantly observe the same property of an object in the environment space.

**Cooperative:** A cooperative sensor network uses the information provided by two independent sensors to derive information that would not be available from the single sensors. An example for a cooperative sensor configuration is stereoscopic vision - by combining two-dimensional images from two cameras at slightly different viewpoints a three-dimensional image of the observed scene is derived. Sensor S4 and S5 in figure 1 represent a cooperative configuration. Both sensors observe the same object, but the measurements are used to form an emerging view on object C that could not have been derived from the measurements of S4 or S5 alone.

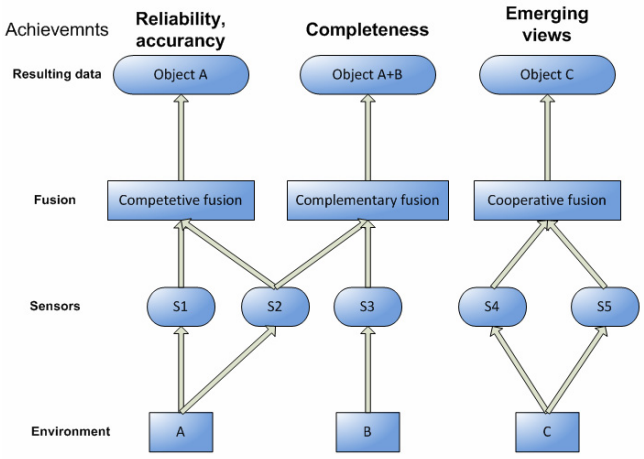


**Fig.1. Competitive, complementary, and cooperative fusion**

These three categories of sensor configuration are not mutually exclusive. Many applications implement aspects of more than one of the three types. An example for such hybrid architecture is the application of multiple cameras that monitor a given area. In regions covered by two or more cameras the sensor configuration can be competitive or cooperative. For regions observed by only one camera the sensor configuration is complementary

## Real –time systems

A real-time system consists of a real-time computer system, a controlled object and an operator.

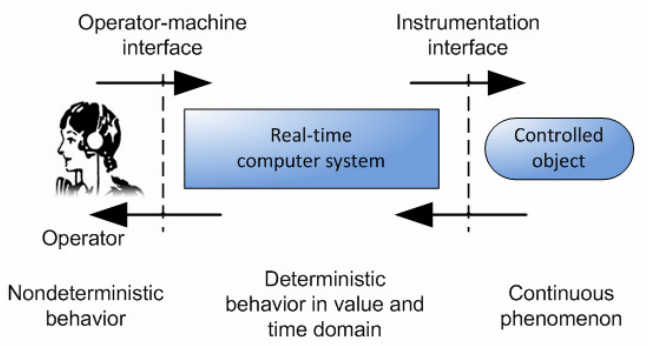


**Fig.2. Parts of a real-time systems**

A real-time computer system is a computer system in which the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical instant at which these results are produced. Figure 2 depicts the parts of a real-time system. The man-machine interface consists of input devices (like keyboards, joysticks, and mouse) and output devices (like displays, alarm lights, loudspeakers) that interface to a human operator.

## Classification of the real-time systems

The distinction is based on the characteristics of the application (e.g., by the consequences of missing a timing requirement or the systems behavior upon failures) and on factors depending on the design and implementation of the real-time computer system (e.g., the method of system activation or assumptions regarding system response times).

### Hard versus Soft real-time systems

Depending on the possible consequences of a missed deadline, hard and soft real-time systems can be distinguished.

**Hard real-time systems** are characterized by the fact that severe consequences will result if logical or timing correctness properties are not satisfied. Hard real-time systems have at least one hard deadline.

**Soft real-time systems** are expected to deliver correct results within specified time intervals, but in contrast to hard real-time systems no severe consequences or catastrophic failures arise from missing those timing requirements.

As an example for a hard real-time system, imagine a fly-by-wire or an anti-lock breaking system that interacts between a pilot or driver and a physical phenomenon.

The requirement on that real-time system is that each user activity is converted to the intended change of the controlled object in the physical environment within a certain time interval. In this scenario an unexpected delay can lead to catastrophic consequences.

### Fail-Safe versus Fail-Operational systems

The reaction of a system upon a critical failure is determined by application requirements. E

Each fault has to be taken as a cause of failure. That means there is inability to serve needed service which the system is expected to serve. It could be the loss of optimal performance, torque, loss of signal, missing message delivery etc. Usual causes of the system failure are faults of the system elements or sensor faults. In these two cases are these failures manifested as inputs integrity failure and should be detected by fusion algorithms or algorithms that are accept-

ing sensor data based on sensing other depended quantities [Vitko03] [Vitko04].

**Fail-safe paradigm:** This model depends on the existence of a safe state that the system can enter upon occurrence of a failure. The existence of such a fail-safe state depends on the application. In fail-safe applications, the real-time computer system must provide high error-detection coverage.

**Fail-operational paradigm:** If a safe state cannot be identified for a given application, the system has to be fail-operational. Fail-operational real-time systems are forced to provide at least a specified minimum level of service for the whole duration of a mission.

An example for a fail-operational real-time system is a flight control system aboard an aero plane. In contrast, a mobile robot operating on the ground usually will be able to quickly enter its safe state by stopping its propulsion, thus representing a fail-safe real-time system.

### Event-Triggered versus Time-Triggered methods

A trigger is an event that initiates some action like the execution of a task or the transmission of a message. The services delivered by a real-time computer system can be triggered in two distinct ways:

**Event-triggered systems:** In an event-triggered system all activities are initiated by the occurrence of events either in the environment or in the real-time computer itself.

**Time-triggered systems:** A time-triggered system derives all the activation points from the progression of physical time.

These two methods can be compared in the time domain, issues of predictability, testability, resource utilization, extensibility, and assumption coverage. Time-triggered systems require an increased effort in the design phase of the system, but provide an easier verification of the temporal correctness. In event-triggered systems, it is generally difficult to make predictions about the system behavior in peak load scenarios.

### Guaranteed Response versus Best Effort

In a hard real-time system, each real-time task must be completed within a prespecified period of time after being requested. If any task fails to complete in time, the entire system fails. In order to validate a hard-real time system, it is required to ensure that all response times will always be met. Depending on the fact if such a promise can be made, systems can be distinguished into:

**Systems with guaranteed response** are validated to hold their specified timing even in case of peak load and fault scenarios. Guaranteed response systems require careful planning and extensive analysis during the design phase.

**Systems with best-effort design** do not require a rigorous specification of load and fault scenarios. It is though very difficult to establish that such a system operates correctly in rare event scenarios.

In contrast to the distinction between hard and soft real-time systems, the difference between guaranteed response and best-effort systems is a property of the real-time computer system and not the real-time application.

## Sensor fusion architectures and applications

Due to the fact that sensor fusion models heavily depend on the application, no generally accepted model of sensor fusion exists until today. It is unlikely that one technique or architecture will provide a uniformly superior solution [KamZhu97]. In this survey, we focus on architectures which have been known for a relatively long period of time.

### The JDL fusion architecture

A frequently referred fusion model originates from the US Joint Directors of Laboratories (JDL). It was proposed in 1985 under the guidance of the Department of Defense. The JDL model comprises five levels of data processing and a database, which are all interconnected by a bus.

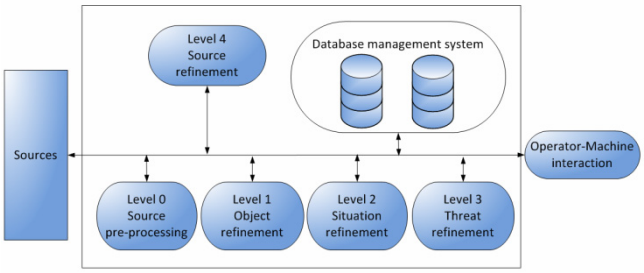


**Fig.3. JDL fusion model**

The five levels are not meant to be processed in a strict order and can also be executed concurrently. Figure 3 depicts the top level of the JDL data fusion process model.

### Waterfall fusion process model

The waterfall model emphasizes on the processing functions on the lower levels. Figure 4 depicts the processing stages of the waterfall model. The stages relate to the levels 0, 1, 2, and 3 of the JDL model as follows: Sensing and signal processing correspond to source preprocessing (level 0), feature extraction and pattern processing match object refinement (level 1), situation assessment is similar to situation refinement (level 2), and decision making corresponds to threat refinement (level 3).

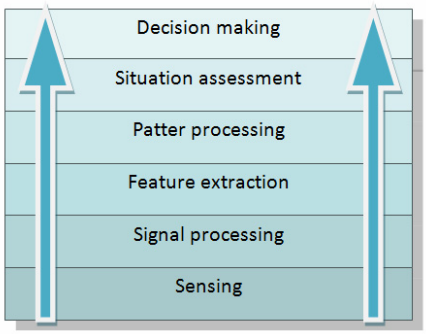


**Fig.4. Waterfall fusion process model**

Being thus very similar to the JDL model, the waterfall model suffers from the same drawbacks. While being more exact in analyzing the fusion process than other models, the major limitation of the waterfall model is the omission of any feedback data flow. The waterfall model has been used in the defense data fusion community in Great Britain, but has not been significantly adopted elsewhere [Bedworth99].

### Boyd model

Boyd has proposed a cycle containing four stages [Boyd87]. This Boyd control cycle or OODA loop (depicted in figure 5) represents the classic decision-support mechanism in military information operations. Because decision-support systems for situational awareness are tightly coupled with fusion systems, the Boyd loop has also been used for sensor fusion.

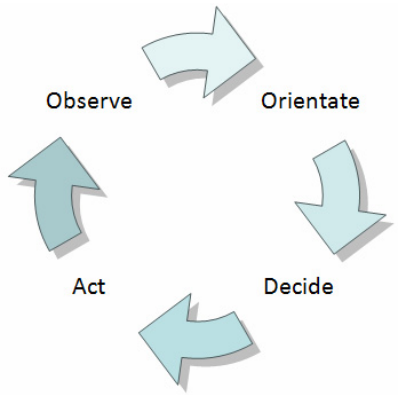


Fig.5. Boyd cycle

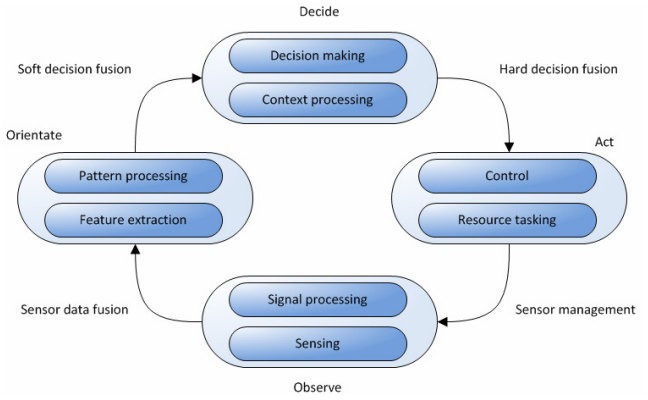


Fig.7. The Omnibus model

### LAAS architecture

LAAS (Laboratoire d'Analyse et d'Architecture des Systémes) architecture was developed as an integrated architecture for the design and implementation of mobile robots with respect to real-time and code reuse. Due to the fact that mobile robot systems often employ sensor fusion methods, the elements of the LAAS architecture are described in the figure 6.

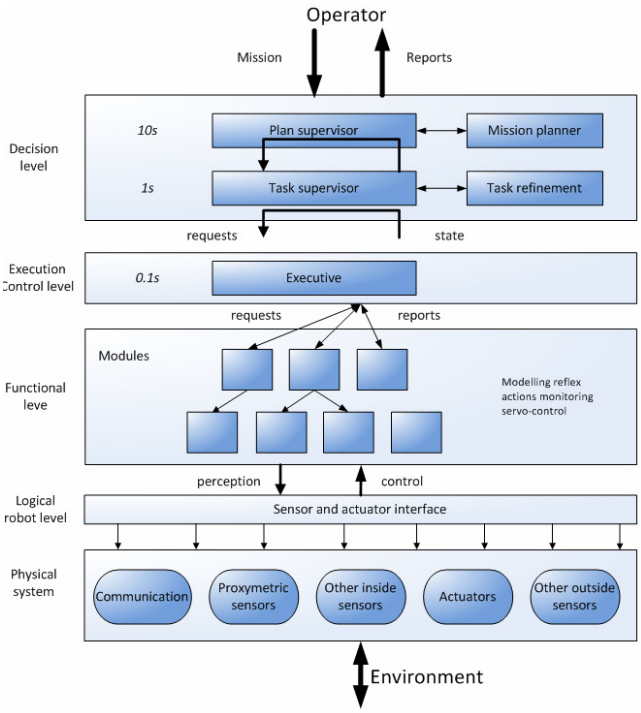


Fig.6. LAAS architecture

### The Omnibus model

The omnibus model [Bedworth99] has been presented in 1999 by Bedworth and O'Brien. Figure 7 depicts the architecture of the omnibus model. Unlike the JDL model, the omnibus model defines the ordering of processes and makes the cyclic nature explicit. It uses a general terminology that does not assume that the applications are defense-oriented. The model shows a cyclic structure comparable to the Boyd loop, but provides a much more fine-grained structuring of the processing levels. The model is intended to be used multiple times in the same application recursively at two different levels of abstraction. First, the model is used to characterize and structure the overall system. Second, the same structures are used to model the single subtasks of the system.

## Conclusion

Applications with certain real-time requirements can be built using various approaches. Due to its highly deterministic behavior, the time-triggered approach is increasingly being recognized as a well-suited basis for building distributed real-time systems. A time-triggered system consists of a set of time-aware nodes. The clocks of all nodes are synchronized in order to establish a global notion of time. Thus, the execution of communication and application tasks takes place at predetermined points in time. Except for the timing, all nodes are independent of each other. This simplifies the replication of services and maintenance tasks.

Therefore, time-triggered architectures also fulfill the dependability requirements for the implementation of fault-tolerant systems using independent redundant components. Additionally, sensor fusion of redundant sensors makes an application more robust to external and internal errors. In case of failures, many sensor fusion algorithms are able to provide a degraded level of service so that the application is able to continue its operation and to provide its service.

Complexity management is supported by sensor fusion as well as by time-triggered distributed systems. Sensor fusion introduces an internal representation of the environmental properties that are observed by sensors. Hence, the control application can be decoupled from the physical sensors, thus improving maintainability and reusability of the code. Moreover, time-triggered architectures support a composable design of real-time applications by breaking up complex systems into small comprehensible components. A system designer introduces interfaces that are well-defined in the value and time domain to each component. Then, all components can be implemented and tested separately. The composability principle takes care of preserving the separately tested functionality of components in the overall application.

The number of sensor fusion algorithms or methods is also numerous - the literature distinguishes filter algorithms (e.g., Kalman Filters), sensor agreement (e.g., voting, sensor selection, fault-tolerant abstract sensors), world-modelling (e.g., occupancy grids, and decision methods (e.g., Bayes inference, Dempster-Shafer reasoning, Fuzzy logic inference). For the information integration these methods represent specific mathematical appliance which help us solve concrete situations and issues where the sensor configuration is applied.

Information integration belongs to nowadays parts of technical progress. Without information integration many systems could not exist which are applied in autonomous or in interactive regimes without collision occurrence. Information integration also shows the way of progress building. There

are still different and difficult problems to solve, thus using and improving of the mathematical appliance is needed for information fusion in the way to guarantee the safety and reliability of operated systems.

*Support provided by grants VEGA 1/3120/06 of The Slovak Ministry of Education is greatly appreciated.*

## References

[ISIF] International Society of Information Fusion (http://www.isif.org/mission.htm)

[Durrant88] H. F. Durrant - Whyte. Sensor Models and Multisensor Integration. International Journal of Robotics Research, 7(6):97113, Dec.1988.

[Vitko03] A. Vitko, M. Šavel, L. Jurišica – Fúzia senzorovej informácie a detekcia/diagnostika porúch v robotike, AT&P Journal 2/2003, Bratislava, Slovensko

[Vitko04] A. Vitko, M. Šavel, L. Jurišica, P. Hubinský – Data fusion and context awareness in autonomous robotics, International journal of Mechanics and Control, Vol.05, No.02,2004, ISSN:1590-8844,pp 29-39

[LuoKay89] R. C. Luo and M. Kay. Multisensor Integration and Fusion in Intelligent Systems. IEEE Transactions on Systems, Man, and Cybernetics, 19(5):901–930, Sep.–Oct. 1989.

[Kopetz05] H. Kopetz, R. Obermaisser, P. Peti - Virtual Networks in an Integrated Time-Triggered Architecture, 2005

[KamZhu97] M. Kam, X. Zhu, and P. Kalata. Sensor Fusion for Mobile Robot Navigation. Proceedings of the IEEE, 85(1):108-119, Jan. 1997.

[Markin97] M. Markin, C. Harris, M. Bernhardt, J. Austin, M. Bedworth, P. Greenway, R. Johnston, A. Little, and D. Lowe. Technology Fore¬sight on Data Fusion and Data Processing. Publication, The Royal Aeronautical Society, 1997.

[Bedworth99] M. D. Bedworth and J. O'Brien. The Omnibus Model: A New Architecture for Data Fusion? In Proceedings of the 2nd Interna¬tional Conference on Information Fusion (FUSION'99), Helsinki, Finnland, July 1999.

[Boyd87] J. R. Boyd. A Discourse on Winning and Losing. Unpublished set of briefing slides, Air University Library, Maxwell AFB, AL, USA, May 1987.

**Ing. Ľubomír Petrovič**
**prof. Ing. Ladislav Jurišica, PhD.**

Slovak University of Technology
Faculty of Electrical Engineering
and Information Technology
Institute of Control and Industrial informatics
Ilkovičova 3
812 19 Bratislava
Email: lubomir.petrovic@stuba.sk