Abbreviations

Note: This annex contains a list of frequently used abbreviations.

AES	Advanced Encryption Standard
ALARP	As Low As Reasonably Practical
API	Application Programming Interface
ASIC	Application Specific Integrated Circuit
AVB	Audio Video Bus
BMTS	Basic Message Transport Service
CAN	Control Area Network
CCF	Concurrency Control Field
EDF	Earliest-Deadline-First
EMI	Electro-Magnetic Interference
EPC	Electronic Product Code
ET	Event-Triggered
FRU	Field-Replaceable Unit
FTU	Fault-Tolerant Unit
GPS	Global Positioning System
IoT	Internet of Things
LIF	Linking Interface
LL	Least-Laxity
MARS	Maintainable Real-Time System
MPSoC	Multiprocessor System on Chip
MSD	Message Structure Declaration
NBW	Non-Blocking Write
NDDC	Non-Deterministic Design Construct
NoC	Network-on-Chip
NTP	Network Time Protocol
PAR	Positive-Acknowledgment-or-Retransmission
PFSM	Periodic Finite State Machine
PIM	Platform Independent Model
PSM	Platform Specific Model
RFID	Radio Frequency Identification
RT	Real-Time
SOC	Sphere of Control
SoC	System on Chip
SRU	Smallest Replaceable Unit
TADL	Task Descriptor List
TAI	International Atomic Time

(continued)

H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Real-Time Systems Series, DOI 10.1007/978-1-4419-8237-7,

TDMA	Time-Division Multiple Access
TMR	Triple-Modular Redundancy
TT	Time Triggered
TTA	Time-Triggered Architecture
TTEthernet	Time-Triggered Ethernet
TTP	Time-Triggered Protocol
UID	Unique Identifier
UTC	Universal Time Coordinated
WCAO	Worst-Case Administrative Overhead
WCCOM	Worst-Case Communication Delay
WCET	Worst-Case Execution Time
WSN	Wireless Sensor Network

Glossary

Note: All terms that are defined in this glossary are put in *italics*. At the end of each entry the section of the book that introduces or discusses the term is mentioned in the parenthesis.

Absolute Timestamp	An *absolute timestamp* of an *event e* is the *timestamp* of this *event* that is generated by the *reference clock* (3.1.2).
Accuracy Interval	The maximum permitted time interval between the *point of observation* of a *real-time entity* and the *point of use* of the corresponding *real-time image* (5.4).
Accuracy of a Clock	The *accuracy* of a clock denotes the maximum offset of a given clock from the external time reference during the time interval of interest (3.1.3).
Action	An *action* is the execution of a program or a communication protocol (1.3.1).
Action Delay	The *action delay* is the maximum time interval between the start of sending a message and the instant when this message becomes *permanent* at the receiver (5.5.1).
Actuator	A *transducer* that accepts data and *trigger* information from a *gateway component* and realizes the intended physical effect in the *controlled object* (9.5).
Advanced Encryption Standard (AES)	An international standard for the encryption of data (6.2.2).
Audio Video Bus (AVB)	The IEEE 802.1 audio/video bridging (AVB) task force develops a set of protocols based on the Ethernet standard that meets the requirements of multimedia systems (7.4.4).
Agreed Data	An *agreed data element* is a *measured data element* that has been checked for plausibility and related to other measured data elements, e.g., by the use of model of the *controlled object*. An agreed data element has been judged to be a correct image of the corresponding real-time entity (➔ *raw data, measured data*) (9.6).

Agreement Protocol	An *agreement protocol* is a protocol that is executed among a set of *components* of a distributed system to come to a common (agreed) view about the state of the world, both in the discrete value domain and in the sparse time domain (9.6).
Alarm Monitoring	*Alarm monitoring* refers to the continuous observation of the *RT entities* to detect an abnormal behavior of the *controlled object* (1.2.1).
Alarm Shower	An *alarm shower* is a correlated set of alarms that is caused by a single *primary event* (1.2.1).
Analytic Rational Subsystem	A conscious human problem-solving subsystem that operates according to the laws of causality and logic (2.1.1).
Anytime Algorithm	An anytime algorithms consist of a *root segment* that calculates a first approximation of the result of sufficient quality and a *periodic segment* that improves the quality of the previously calculated result. The periodic segment is executed repeatedly until the deadline is reached (10.2.3).
Aperiodic Task	An *aperiodic task* is a *task* where neither the *task request times* nor the minimum time interval between successive requests for activation are known (➔ *periodic task,* ➔ *sporadic task*) (10.1.2).
Application Programming Interface (API)	The interface between an application program and the operating system within a component (9.1.4).
A Priori Knowledge	Knowledge about the future behavior of a system that is available ahead of time (1.5.5).
ARINC 629 Protocol	A medium access protocol that controls access to a single communication channel by a set of components. It is based on a set of carefully selected time-outs (7.4.2).
Assumption Coverage	*Assumption coverage* is the probability that assumptions that are made in the model building process hold in reality. The *assumption coverage* limits the probability that conclusions derived from a perfect model will be valid in the real world (1.5.3).
Atomic Action	An *atomic action* is an action that has the all-or-nothing property. It either completes and delivers the intended result or does not have any effect on its environment (4.2.3).
Atomic Data Structure	An *atomic data structure* is a data structure that has to be interpreted as a whole (4.3).
Availability	*Availability* is a measure of the correct service delivery regarding the alternation of correct and incorrect service, measured by the fraction of time that the system is ready to provide the service (1.4.4).

Babbling Idiot	A *component* of a distributed computer system that sends messages outside the specified time interval is called a *babbling idiot* (4.7.1).
Back-Pressure Flow Control	In *back-pressure flow control* the receiver of a sequence of messages exerts back pressure on the sender so that the sender will not outpace the receiver (7.3.2).
Basic Message Transport Service (BMTS)	The *basic message transport service* transports a *message* from a sending component to one or more receiving components (7.2.1).
Benign Failure	A *failure* is *benign* if the worst-case failure costs are of the same order of magnitude as the loss of the normal utility of the system (6.1.3).
Best Effort	A *real-time system* is a *best-effort* system if it is not possible to establish the temporal properties by analytical methods, even if the *load- and fault hypothesis* holds (➔ guaranteed timeliness) (1.5.3).
Bit-length of a Channel	The *bit length of a channel* denotes the number of bits that can traverse the channel within one *propagation delay* (7.2.2).
Bus Guardian	The independent hardware unit of a TTP controller that ensures *fail silence* in the temporal domain (7.5.1).
Byzantine Error	A *Byzantine error* occurs if a set of receivers observes different (conflicting) values of a *RT entity*. Some or all of these values are incorrect (synonym: malicious error, two-faced error, inconsistent error) (3.4.1).
Causal Order	A *causal order* among a set of *events* is an order that reflects the cause-effect relationships between the *events* (3.1.1).
Causality	The *causality* relationship between a cause C and an event E is defined as follows: If C happens, then E is always produced by it (2.1.1).
Clock	A *clock* is a device for time measurement that contains a counter and a physical oscillation mechanism that periodically generates an *event,* the ➔ *tick* or ➔ *microtick* of the clock, to increase the counter (3.1.2).
Cluster	A *cluster* is a subsystem of a real-time system. Examples of clusters are the *real-time computer system*, the operator, or the *controlled object* (1.1).
Cognitive Complexity	The elapsed time needed to ➔ *understand* a model by a given observer is a measure for the cognitive effort and thus for the cognitive complexity of a model relative to the observer. We assume that the given observer is representative for the intended user group of the model.

Complex Task (C-task)	A *complex task (C-task)* is a *task* that contains a blocking synchronization statement (e.g., a semaphore operation *wait*) within the *task* body (9.2.3).
Component	A *component* is a hardware-software unit, i.e., a self-contained computer including system- and application software that performs a well-defined function within a distributed computer system (4.1.1).
Composability	An architecture is *composable* regarding a specified property if the system integration will not invalidate this property, provided it has been established at the subsystem level (4.7.1).
Computational Cluster	A subsystem of a real-time system that consists of a set of *components* interconnected by a *real-time communication network* (1.1).
Concept	A *concept* is a category that is augmented by a *set of beliefs* about its relations to other categories. The set of beliefs relates a *new concept* to already *existing concepts* and provides for an *implicit theory* (2.1.2).
Conceptual Landscape	The *conceptual landscape* refers to the *personal knowledge base* that has been built up and maintained by an individual in the *experiential* and *rational* subsystem of the mind (2.2).
Concrete World Interface	The *concrete world interface* is the physical I/O *interface* between an *interface component* and an external device or another external component (4.5).
Concurrency Control Field (CCF)	The *concurrency control field (CCF)* is a single-word data field that is used in the *NBW protocol* (9.4.2).
Consistent Failure	A *consistent failure* occurs if all users see the same erroneous result in a multi-user system (6.1.3).
Contact Bounce	The random oscillation of a mechanical contact immediately after closing (9.5.2).
Control Area Network (CAN)	The *control area network (CAN)* is a low-cost *event*-triggered communication network that is based on the carrier-sense multiple-access collision-avoidance technology (7.3.2).
Controlled Object	The *controlled object* is the industrial plant, the process, or the device that is to be controlled by the *real-time computer system* (1.1).
Convergence Function	The *convergence function* denotes the maximum *offset* of the local representations of the global time within an ensemble of *clocks* (3.4).
Deadline	A *deadline* is the instant when a result should/must be produced (➔ *soft deadline, firm deadline, and hard deadline*) (1.1).

Deadline Interval	The *deadline interval* is the interval between the *task request time* and the *deadline* (10.1).
Determinism	A physical system behaves deterministically if given an initial state at instant t and a set of future timed inputs, then the future states and the values and times of future outputs are entailed. In a deterministic distributed computer system, we must assume that all events, e.g., the observation of the initial state at instant t and the timed inputs, are *sparse events* on a *sparse* global time base (5.6.1).
Drift	The *drift* of a physical *clock k* between *microtick i* and *microtick i+1* is the frequency ratio between this *clock k* and the *reference clock* at the time of *microtick i*. (3.1.2).
Drift Offset	The *drift offset* denotes the maximum deviation between any two good *clocks* if they are free running during the resynchronization interval (3.1.4).
Duration	A *duration* is a section of the timeline (3.1.1).
Dynamic Scheduler	A *dynamic scheduler* is a *scheduler* that decides at run time after the occurrence of a significant *event* which *task* is to be executed next (10.4).
Earliest-Deadline-First (EDF) Algorithm	An optimal dynamic preemptive scheduling algorithm for scheduling a set of independent *tasks* (10.4.1).
Electro-Magnetic Interference (EMI)	The disturbance of an electronic system by electromagnetic radiation (11.3.4).
Electronic Product Code (EPC)	A code designed by the RFID community that can be used to uniquely identify every product on the globe (13.4.2).
Embedded System	A *real-time computer* that is embedded in a well specified larger system, consisting in addition to the embedded computer of a mechanical subsystem and, often, a man-machine *interface* (➔ *intelligent product*) (1.6.1).
Emergence	We speak of *emergence* when the interactions of subsystems give rise to unique global properties at the system level that are not present at the level of the subsystems.
End-to-End Protocol	An *end-to-end protocol* is a protocol between the users (machines or humans) residing at the end points of a communication channel (1.7).
Environment of a Computational Cluster	The *environment* of a given *computational cluster* is the set of all *clusters* that interact with this *cluster*, either directly or indirectly (1.1).
Error	An *error* is that part of the state of a system that deviates from the intended specification (6.1.2).
Error-Containment Coverage	Probability that an *error* that occurs in an *error-containment region* is detected at one of the *interfaces* of this region (6.4.2).

Error-Containment Region	A subsystem of a computer system that is encapsulated by error-detection *interfaces* such that the there is a high probability (the ➔ *error containment coverage*) that the consequences of an *error* that occurs within this subsystem will not propagate outside this subsystem without being detected (6.4.2).
Event	An *event* is a happening at a cut of the time-line. Every change of state is an *event* (1.1).
Event Message	A message is an *event message* if it contains information about events and if every new version of the message is queued at the receiver and consumed on reading (➔ *state message*) (4.3.3).
Event-triggered (ET) Observation	An *observation* is *event-triggered* if the *point of observation* is determined by the occurrence of an *event* other than a *tick* of a *clock* (5.2).
Event-Triggered (ET) System	A *real-time computer system* is *event-triggered* (ET) if all communication and processing activities are triggered by *events* other than a clock *tick* (1.5.5).
Exact Voting	A *voter that* considers two messages the same if they contain the exactly same sequence of bits (➔ inexact voter) (6.4.2).
Execution Time	The *execution time* is the *duration* it takes to execute an *action* by a computer. If the speed of the oscillator that drives a computer is increased, the execution time is decreased. The *worst-case execution time* is called ➔ *WCET* (4.1.2).
Explicit Flow Control	In *explicit flow control* the receiver of a message sends an explicit acknowledgment message to the sender, informing the sender that the previously sent message has correctly arrived and that the receiver is now ready to accept the next message (➔ *flow control,* ➔ *implicit flow control*) (7.2.3).
External Clock Synchronization	The process of synchronization of a *clock* with a *reference clock* (3.1.3).
Fail-Operational System	A *fail-operational system* is a *real-time system* where a safe state cannot be reached immediately after the occurrence of a *failure* (1.5.2).
Fail-Safe System	A *fail-safe system* is a *real-time system* where a safe state can be identified and quickly reached after the occurrence of a *failure* (1.5.2).
Fail-Silence	A subsystem is *fail-silent* if it either produces correct results or no results at all, i.e., it is quiet in case it cannot deliver the correct service (6.1.1).
Failure	A *failure* is an *event* that denotes a deviation of the actual service from the intended service (6.1.3).

Fault	A *fault* is the cause of an *error* (6.1.1).
Fault Hypothesis	The *fault hypothesis* identifies the assumptions that relate to the type and frequency of faults that a fault-tolerant computer system is supposed to handle (6.1.1).
Fault-Tolerant Average Algorithm (FTA)	A distributed clock synchronization algorithm that handles *Byzantine* failures of *clocks* (3.4.3).
Fault-Containment Unit (FCU)	A unit that contains the direct consequences of a fault. Different FCUs must fail independently. A *component* should be an FCU. (6.4.2).
Fault-Tolerant Unit (FTU)	A unit consisting of a number of replica determinate ➔ FCUs that provides the specified service even if some of its constituent FCUs *(components)* fail (6.4.2).
Field Replaceable Unit (FRU)	An FRU is a subsystem that is considered atomic from the point of view of a repair action (1.4.3).
Firm Deadline	A *deadline* for a result is *firm* if the result has no utility after the deadline has passed (1.1).
FIT	A FIT is a unit for expressing the failure rate. 1 FIT is 1 failure/10^{-9} h (1.4.1).
Flow Control	*Flow control* assures that the speed of the information flow between a sender and a receiver is such that the receiver can keep up with the sender (➔ *explicit flow control*, ➔ *implicit flow control*) (7.2.3).
Gateway component	A *component* of a distributed real-time system that is a member of two *clusters* and implements the relative views of these two interacting *clusters* (4.5).
Global Time	The *global time* is an abstract notion that is approximated by a properly selected subset *of the microticks* of each synchronized local clock of an ensemble. The selected *microticks* of a local *clock* are called the *ticks* of the *global time* (3.2.1).
Granularity of a Clock	The *granularity* of a *clock* is the nominal number of *microticks* of the *reference clock* between two *microticks* of the *clock* (3.1.2).
Ground (g) State	The *ground state* of a *component* of a distributed system at a given level of abstraction is a *state* at an instant where there is a minimal dependency of future behavior on past behavior. At the ground state instant all information of the past that is considered relevant for the future behavior is contained in a declared ground state data structure. At the ground state instant no *task* is active and all communication channels are flushed. The instants of the ground state are ideal for reintegrating components (4.2.3).
Guaranteed Timeliness	A *real-time system* is a *guaranteed timeliness* system if it is possible to reason about the temporal adequacy of

	the design without reference to probabilistic arguments, provided the assumptions about the *load- and fault hypothesis* hold (➔ *best effort*) (1.5.3).
Hamming Distance	The *Hamming distance* is one plus the maximum number of bit errors in a codeword that can be detected by syntactic means (6.3.3).
Hard Deadline	A *deadline* for a result is *hard* if a catastrophe can occur in case the deadline is missed (1.1).
Hard Real-Time Computer System	A *real-time computer system* that must meet at least one *hard deadline* (Synonym: *safety-critical real-time computer system*) (1.1).
Hazard	A *hazard* is an undesirable condition that has the potential to cause or contribute to an accident (11.4.2).
Hidden Channel	A communication channel outside the given *computational cluster* (5.5.1).
Idempotency	*Idempotency* is a relation between a set of replicated messages arriving at the same receiver. A set of replicated messages is *idempotent* if the effect of receiving more than one copy of a message is the same as receiving only a single copy (5.5.4).
Implicit Flow Control	In *implicit flow control*, the sender and receiver agree *a priori,* i.e., before the start of a communication session, about the instants when messages will be sent. The sender commits to send only messages at the agreed instants, and the receiver commits to accept all messages sent by the sender, as long as the sender fulfills its obligation (➔ *explicit flow control,* ➔ *flow control*) (7.2.3).
Inexact Voting	A *voter* that considers two messages the "same" if both of them conform to some application specific "sameness" criterion (➔ *exact voter*) (6.4.2).
Instant	An *instant* is a cut of the timeline (1.1).
Instrumentation Interface	The *instrumentation interface* is the *interface* between the *real-time computer system* and the *controlled object* (1.1).
Intelligent Actuator	An *intelligent actuator* consists of an actuator and a microcontroller, both mounted together in a single housing (9.5.5).
Intelligent Product	An *intelligent product* is a self-contained system that consists of a mechanical subsystem, a user *interface,* and a controlling embedded *real-time computer system* (➔ *embedded system*) (1.6.1).
Intelligent Sensor	An *intelligent sensor* consists of a sensor and a microcontroller such that *measured data* is produced at the output *interface*. If the *intelligent sensor* is fault-tolerant, *agreed data* is produced at the output *interface* (9.5.5).

Interface	An *interface* is a common boundary between two subsystems (4.4).
Interface Component	A *component* with an *interface* to the external environment of a component. An *interface component* is a *gateway* (4.5).
Internal Clock Synchronization	The process of mutual synchronization of an ensemble of *clocks* in order to establish a *global time* with a bounded *precision* (3.1.3).
International Atomic Time (TAI)	An international time standard, where the second is defined as 9 192 631 770 periods of oscillation of a specified transition of the Cesium atom 133 (3.1.4).
Intrusion	the successful exploitation of a *vulnerability* (6.2).
Intuitive Experiental Problem Solving System	A human preconscious emotionally-based problem-solving subsystem that operates holistically, automatically, and rapidly, and demands minimal cognitive resources for its execution (2.1.1).
Internet of Things (IoT)	The direct connection of physical things to the Internet such that remote access and control of physical devices is enabled (13).
Irrevocable action	An action that cannot be undone, e.g., drilling a hole, activation of the firing mechanism of a firearm (1.5.1).
Jitter	The *jitter* is the difference between the maximum and the minimum duration of an *action* (processing action, communication action) (1.3.1).
Laxity	The *laxity* of a *task* is the difference between the *deadline interval* minus the *execution time* (the *WCET*) of the *task* (9.2.2).
Least-Laxity (LL) Algorithm	An optimal dynamic preemptive *scheduling* algorithm for scheduling a set of independent *tasks* (10.4.1).
Logical Control	*Logical control* is concerned with the control flow *within* a *task*. The *logical control* is determined by the given program structure and the particular input data to achieve the desired data transformation (➔ *temporal control*) (4.1.3).
Maintainability	The *Maintainability (d)* is the probability that the system is restored to its operational state and restarted within a time interval *d* after a failure (1.4.3).
Malicious Code Attack	A malicious code attack is an attack where an adversary inserts malicious code, e.g., a virus, a worm, or a Trojan horse, into the software in order that the attacker gets partial or full control over the system (6.2.2).
Measured Data	A *measured data element* is a *raw data element* that has been preprocessed and converted to standard technical units. A sensor that delivers *measured data* is called an *intelligent sensor* (➔ *raw data, agreed data*) (9.6.1).

Membership Service	A *membership service* is a service in a distributed system that generates consistent information about the operational state (operating or failed) of all *components* at agreed instants (membership points). The length of the interval between a membership point and the moment when the consistent membership information is available at the other *components* is a quality of service parameter of the membership service (5.3.2).
Message Structure Declaration (MSD)	A specification that explains how the data field of a message is structured into syntactic units and assigns names to these syntactic units. The names identify the *concepts* that explain the meaning of the data (4.6.2).
Microtick	A *microtick* of a physical clock is a periodic *event* generated by this *clock* (➔ *tick*) (3.1.2).
Non-Blocking Write Protocol (NBW)	The *non-blocking write protocol* (*NBW*) is a synchronization protocol between a single writing task and many reading tasks that achieves data consistency without blocking the writer (9.4.2).
Non-Deterministic Design Construct (NDDC)	A non-deterministic design construct is a design construct that produces unpredictable result either in the value domain or the temporal domain (5.6.3).
Observation	An observation of a real-time entity is an atomic triple consisting of the name of the real-time entity, the instant of the observation, and the value of the real-time entity (5.2).
Offset	The offset between two *events* denotes the time difference between these *events* (3.1.3).
Periodic Finite State Machine (PFSM)	A PFSM is an extension of the finite state machine model to include the progression of real time (4.1.3).
Periodic Task	A *periodic task* is a *task* that has a constant time interval between successive *task* request times (➔ *aperiodic task,* ➔ *sporadic task*) (10.1.2).
Permanence	*Permanence* is a relation between a given message and all related messages that have been sent to the same receiver before this given message has been sent. A particular message becomes *permanent* at a given *component* at the moment when it is known that all earlier sent related messages have arrived (or will never arrive) (5.5.1).
Phase-Aligned Transaction	A *phase-aligned transaction* is a *real-time transaction* where the constituting processing and communication *actions* are synchronized (5.4.1).
Point of Observation	The instant when a *real-time entity* is observed (1.2.1).

Positive-Acknowledgment-or-Retransmission (PAR) protocol	The *Positive-Acknowledgment-or-Retransmission (PAR) protocol* is an *event*-triggered protocol where a message sent by the sender must be positively acknowledged by the receiver (7.1.2).
Precision	The *precision* of an ensemble of clocks denotes the maximum *offset* of respective ticks of any two clocks of the ensemble over the period of interest. The *precision* is expressed in the number of *ticks* of the *reference clock* (3.1.3).
Primary Event	A *primary event* is the cause of an *alarm shower* (1.2.1).
Priority Ceiling Protocol	A *scheduling* algorithm for *scheduling* a set of dependent periodic *tasks* (10.4.2).
Process Lag	The delay between applying a step function to an input of a *controlled object* and the start of response of the *controlled object* (1.3.1).
Propagation Delay	The *propagation delay* of a communication channel denotes the time interval it takes for a single bit to traverse the channel (7.2.2).
Protocol	A *protocol* is a set of rules that governs the communication among partners (2.2.3).
Radio Frequency Identification (RFID)	A technology for the identification of objects by electronic means (13.4)
Rare Event	A *rare event* is a seldomly occurring event that is of critical importance. In a number of applications the predictable performance of a *real-time computer system* in *rare event* situations is of overriding concern (1.2.1).
Rate-Monotonic Algorithm	A dynamic preemptive *scheduling* algorithm for *scheduling* a set of independent periodic *tasks* (10.4.1).
Raw Data	A *raw data element* is an analog or digital data element as it is delivered by an unintelligent sensor (➔ *measured data, agreed data*) (9.6.1).
Real-Time (RT) Entity	A *real-time (RT) entity* is a state variable, either in the *environment* of the *computational cluster,* or in the *computational cluster* itself, that is relevant for the given purpose. Examples of *RT entities* are: the temperature of a vessel, the position of a switch, the setpoint selected by an operator, or the intended valve position calculated by the computer (5.1).
Real-Time (RT) Image	A *real-time (RT) image* is a current picture of a *real-time entity* (5.3).
Real-Time Computer System	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 time when these results are produced. A real-time computer system can consist of one or more *computational clusters* (1.1).

Real-time Data Base	The *real-time database* is formed by the set of all *temporally accurate real-time images* (1.2.1).
Real-Time Object	A *real-time (RT) object* is a container inside a computer for a *RT entity* or a *RT image*. A *clock* with a granularity that is in agreement with the dynamics of the *RT object* is associated with every RT object (5.3.2).
Real-Time Transaction	A *real-time (RT) transaction* is a sequence of computational and communication *actions* between a stimulus from the environment and a response to the environment of a *computational cluster* (1.7.3).
Reasonableness Condition	The *reasonableness condition* of clock synchronization states that the *granularity* of the *global time* must be larger than the *precision* of the ensemble of *clocks* (3.2.1).
Reference Clock	The *reference clock* is an ideal *clock* that ticks always in perfect agreement with the international standard of time (3.1.2).
Reliability	The *reliability R (t)* of a system is the probability that a system will provide the specified service until time *t*, given that the system was operational at $t = t_o$ (1.4.1).
Replica Determinism	*Replica Determinism* is a desired relation between replicated *RT objects*. A set of replicated *RT objects* is *replica determinate* if all objects of this set have the same visible *state* and produce the same output messages at instants that are at most an interval of *d* time units apart (5.6).
Resource Adequacy	A *real-time computer system* is *resource adequate* if there are enough computing resources available to handle the specified *peak load* and the *faults* specified in the *fault hypothesis*. Guaranteed response systems must be based on *resource adequacy (➔ guaranteed timeliness)* (1.5.4).
Rise Time	The *rise time* is the time required for the output of a system to rise to a specific percentage of its final equilibrium value as a result of step change on the input (1.3.1).
Risk	*Risk* is the product of *hazard* severity and *hazard* probability. The severity of a *hazard* is the worst-case damage of a potential accident related to the *hazard* (11.4.2).
Safety	*Safety* is *reliability* regarding *critical failure* modes (1.4.2).
Safety Case	A *safety case* is a combination of a sound set of arguments supported by analytical and experimental evidence substantiating the *safety* of a given system (11.4.3).

Safety Critical Real-Time Computer System	Synonym to *hard real-time computer system* (1.1).
Sampling	In *sampling*, the state of a RT entity is periodically interrogated by the computer system at instants that are in the *sphere of control* of the computer system. If a memory element is required to store the effect of an *event*, the memory element is outside the *sphere of control* of the computer system (1.3.1).
Schedulability Test	A *schedulability test* determines whether there exists a schedule such that all *tasks* of a given set will meet their deadlines (10.1.1).
Semantic Agreement	An agreement among measured variables is called *semantic agreement* if the meanings of the different *measured values* are related to each other by a process model that is based on *a priori* knowledge about the physical characteristics and the dynamics of the *controlled object* (9.6.3).
Semantic Content	The essential meaning of a statement or variable as understood by an end-user. The same *semantic content* can be represented in different syntactic forms (2.2.4).
Signal Conditioning	*Signal conditioning* refers to all processing steps that are required to generate a *measured data element* from a *raw data element* (1.2.1).
Soft Deadline	A *deadline* for a result is *soft* if the result has utility even after the *deadline* has passed (1.1).
Soft Real-Time Computer System	A *real-time computer system* that concerned with any *soft deadlines* only (1.1).
Sparse Event	an event that occurs in the active interval of a ➔ sparse time base (3.3).
Sparse Time Base	a time-base in a distributed computer systems where the physical time is partitioned into an infinite sequence of active and silent intervals and where *sparse events* may be generated only in the active intervals (3.3).
Sphere of Control (SOC)	The *sphere of control* of a subsystem is defined by the set of *RT entities* the values of which are established within this subsystem (5.1.1).
Sporadic Task	A *sporadic task* is a *task* where the *task* request times are not known but where it is known that a minimum time interval exists between successive requests for execution (➔ *periodic task,* ➔ *aperiodic task*) (10.1.2).
Spoofing Attack	A security attack where an adversary masquerades as a legitimate user in order to gain unauthorized access to a system (6.2.2).
State	The state of a component at a given instant is a data structure that contains all information about the past that

	is considered relevant for the future operation of the component (4.2).
State Estimation	*State estimation* is the technique of building a model of a *RT entity* inside a *RT object* to compute the probable state of a *RT entity* at a selected future instant, and to update the related *RT image* accordingly (5.4.3).
State Message	A message is a *state message* if it contains information about states, if a new version of the message replaces the previous version, and the message is not consumed on reading (➔ *event message*) (4.3.4).
Synchronization Condition	The *synchronization condition* is a necessary condition for the synchronization of clocks. It relates the *convergence function,* the *drift offset* and the *precision* (3.4.1).
System of Systems (SoS)	A system consisting of a set of nearly autonomous constituent systems that decide to cooperate in order to achieve a common objective (4.7.3).
Task Descriptor List (TADL)	The *task* descriptor list (TADL) is a static data structure in a time-triggered operating system that contains the instants when the *tasks* have to be dispatched (9.2.1).
Task Request Time	The *task request time* is the instant when a *task* becomes ready for execution (10.1.2).
Task	A *task* is the execution of a program (➔ *simple task,* ➔ *complex task*) (4.2.1).
Temporal Accuracy	A *real-time image* is *temporally accurate* if the time interval between the moment "now" and instant when the current value of the real-time image was the value of the corresponding *RT entity* is smaller than an application specific bound (5.4).
Temporal Control	*Temporal control* is concerned with the determination of the real-time instants when a *task* must be activated or when a *task* must be blocked (➔ *logical control*) (4.1.3).
Temporal Failure	A *temporal failure* occurs when a value is presented at the system-user *interface* outside the intended interval of real-time. Temporal failures can only exist if the system specification contains information about the expected temporal behavior of the system (Synonym timing failure) (6.1.3).
Temporal Order	The *temporal order* of a set of *events* is the order of *events* as they occurred on the time line (3.1.1).
Thrashing	The phenomenon that a system's throughput decreases abruptly with increasing load is called *thrashing* (7.2.4).
Tick	A *tick* (synonym: macrotick) of the global time is a selected *microtick* of the local clock. The *offset* between any two respective global ticks of an ensemble of synchronized *clocks* must always be less than the *precision*

	of the ensemble (➔ *microtick, reasonableness condition*) (3.2.1).
Time Stamp	A *timestamp* of an *event* with respect to a given clock is the state of the clock at the instant of occurrence of the *event* (3.1.2).
Time-Division Multiple Access (TDMA)	*Time-Division Multiple Access* is a time-triggered communication technology where the time axis is statically partitioned into slots. Each slot is statically assigned to a *component*. A *component* is only allowed to send a message during its slot (7.5).
Time-Triggered Architecture (TTA)	A distributed computer architecture for real-time applications, where all components are aware of the progression of the global time and where most actions are triggered by the progression of this global time.
Time-Triggered Ethernet (TTEthernet)	An extension of standard Ethernet that supports deterministic message transport (7.5.2).
Time-Triggered Protocol (TTP)	A communication protocol where the instant of starting a message transmission is derived from the progression of the global time (7.5.1).
Timed Message	A *timed message* is a message that contains the timestamp of an *event* (e.g., point of observation) in the data field of the message (9.1.1).
Timing Failure	➔ Temporal Failure
Token Protocol	A communication protocol where the right to transmit is contained in a token that is passed among the communicating partners (7.4.1).
Transducer	A device converting energy from one domain into another. The device can either be a *sensor* or an *actuator* (9.5).
Transient Fault	A *transient fault* is a fault that exists only for a short period of time after which it disappears. The hardware is not permanently affected by a transient fault (6.1.1).
Trigger	A *trigger* is an *event* that causes the start of some action (1.5.5).
Trigger Task	A *trigger task* is a time-triggered *task* that evaluates a condition on a set of temporally accurate variables and generates a *trigger* for an application *task* (9.2.2).
Triple-Modular Redundancy (TMR)	A fault-tolerant system configuration where a *fault-tolerant unit (FTU)* consists of three synchronized replica deterministic *components*. A value or timing failure of one *component* can be masked by the majority (➔ *voting*) (6.4.2).
Understanding	*Understanding* develops if the concepts and relationships that are employed in the representation a model have been adequately linked with the ➔ *conceptual*

	landscape and the methods of reasoning of the observer (2.1.3).
Universal Time Coordinated (UTC)	An international time standard that is based on astronomical phenomena (➔ *International Atomic Time*) (3.1.4).
Value Failure	A *value failure* occurs if an incorrect value is presented at the system-user *interface* (6.1.3).
Voter	A *voter* is a unit that detects and masks errors by comparing a number of independently computed input messages and delivers an output message that is based on the analysis of the inputs (➔ *exact voting*, ➔ *inexact voting*) (6.4.2).
Vulnerability	A *deficiency* in the design or operation of a computer system that can lead to a security incident, such as an *intrusion* (6.2).
Watchdog	A *watchdog* is an independent external device that monitors the operation of a computer. The computer must send a periodic signal (*life sign*) to the *watchdog*. If this life sign fails to arrive at the *watchdog* within the specified time interval, the *watchdog* assumes that the computer has failed and takes some action (e.g., the *watchdog* forces the *controlled object* into the safe state) (9.7.4).
Worst-Case Administrative Overhead (WCAO)	The *worst-case execution time* of the administrative services provided by an operating system (5.4.2).
Worst-Case Communication Delay (WCCOM)	The *worst-case communication delay* is the maximum duration it may take to complete a communication action under the stated *load- and fault hypothesis* (5.4.1).
Worst-Case Execution Time (WCET)	The *worst-case execution time (WCET)* is the maximum duration it may take to complete an *action* under the stated *load- and fault hypothesis*, quantified over all possible input data (10.2).

References

[Ahu90] Ahuja, M., Kshemkalyani, A. D. & Carlson, T. (1990). *A Basic Unit of Computation in a Distributed System*. Proc. of the 10th IEEE Distributed Computer Systems Conference. IEEE Press. (pp. 12-19).

[Ale77] Alexander, C.S. et al. (1977). *A Pattern Language*. Oxford University Press.

[Ami01] Amidzic, O., H.J.Riehle, T. Fehr, C. Wienbruch &T. Elbert. (2001). *Pattern of Focal y-bursts in Chess Players*. Nature. Vol. 412. (p. 603).

[And01] Anderson, D.L. (2001). *Occam's Razor; Simplicity, Complexity, and Global Geodynamics*. Proc. of the American Philosophical Society. Vol.14(1). (pp. 56-76).

[And95] Anderson, J., S. Ramamurthy, & K. Jeffay. (1995). *Real-Time Computing with Lock-Free Shared Objects*. Proc. RTSS 1995. IEEE Press. (pp. 28-37).

[ARI05] ARINC. (2005). *Design Assurance Guidance for airborne electronic hardware* RTCA/DO-254. ARINC, Annapolis, Maryland.

[ARI91] ARINC. (1991). *Multi-Transmitter Data Bus ARINC 629–Part 1: Technical Description*. ARINC, Annapolis, Maryland.

[ARI92] ARINC. (1992). *Software Considerations in Airborne Systems and Equipment Certification* ARINC DO-178B. ARINC, Annapolis, Maryland.

[Arl03] Arlat, J. et al. (2003). *Comparison of Physical and Software-Implemented Fault Injection Techniques*. IEEE Trans. on Computers. Vol. 52(9). (pp. 1115-1133).

[Art06] ARTEMIS. (2006). *Strategic Research Agenda. Reference designs and architectures*. URL: https://www.artemisia-association.org/downloads/RAPPORT_RDA.pdf

[Att09] Attaway, S. (2009). *Matlab, a Practical Introduction to Programming and Problem Solving*. Elsevier.

[Avi04] Avizienis, A., et al., (2004). *Basic concepts and taxonomy of dependable and secure computing*. IEEE Trans. on Dependable and Secure Computing. Vol. 1(1). (pp. 11-33).

[Avi82] Avizienis, A. (1982). *The Four-Universe Information System Model for the Study of Fault Tolerance*. Proc. of the 12th FTCS Symposium. IEEE Press. (pp. 6-13).

[Avi85] Avizienis, A. (1985). *The N-version Approach to Fault-Tolerant Systems*. IEEE Trans. on Software Engineering. Vol. 11(12). (pp. 1491-1501).

[Avr92] Aversky, D., Arlat, J., Crouzet, Y., & Laprie, J. C. (1992). *Fault Injection for the Formal Testing of Fault Tolerance*. Proc. of FTCS 22. IEEE Press. (pp. 345-354).

[Bar01] Baresi, L. and M. Young. (2001). *Test Oracles*. University of Oregon, Dept. of Computer Science.

[Bar07] Baronti, P., et al. (2007). *Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and Zigbee Standards*. Computer Communication, Vol. 30. Elsevier. (pp. 1655-1695).

[Bar93] Barborak, M., Malek, M. (1993). The Consensus Problem in Fault-Tolerant Computing. ACM Computing Surveys. Vol 25(2). (pp. 171-218).

[Bea09] Beautement, A., M.A. Sasse, & M. Wonham. (2009). *The Compliance Budget: Managing Security Behavior in Organizations*. Proc of NSPW 08. ACM Press. (pp. 47-58).

[Bed08] Bedau, M.A. & P. Humphrey. (2008). *Emergence*. MIT Press, Cambridge.

[Ben00] Benini, L. & G. DeMicheli. (2000). *System Level Power Estimation: Techniques and Tools*. ACM Trans. on Design Automation of Electronic Systems. Vol. 5(2). (pp. 115-192).

[Ber01] [Ber01] Berwanger, J., et al. (2001). *FlexRay–The Communication System for Advanced Automotive Control Systems*. SAE Transactions, Vol. 110(7). SAE Press. (pp. 303-314).

[Ber07] Bertolino, A. (2007). *Software Testing Research: Achievements, Challenges, Dreams*. Proc. of FOSE 07. IEEE Press. (pp. 85-103).

[Ber85] Berry, G. & L. Cosserat. (1985). *The Synchronous Programming Language ESTEREL and its Mathematical Semantics*. Proc. of the Seminar on Concurrency. LNCS 197. Springer-Verlag.

[Bha10] Bhattacharayya, R. et al. (2010). *Low-Cost, Ubiquitous RFID-Tag-Antenna-Based Sensing*. Proc. of the IEEE. Vol. 98(10). (pp. 1593-1600).

[Bla09] Black, D.C., J. Donovan, & B. Bunton. (2009). *System C: From the Ground Up*. Springer Verlag.

[Boe01] Boehm, B. & V. Basili. (2001). *Software Defect Reduction Top 10 List*. IEEE Computer. January 2001. (pp. 135-137).

[Bor07] Borkar, S. (2007). *Thousand Core Chips–a Technology Perspective*. Proc. of DAC 2007. ACM Press. (pp. 746-749).

[Bou61] Boulding, K.E. (1961). *The Image*. Ann Arbor Paperbacks.

[Bou96] Boussinot, F. & R. Simone. (1996). *The SL Synchronous Language*. IEEE Trans. on Software Engineering. Vol. 22(4). (pp. 256-266).

[Bro00] Brown, S. (2000). *Overview of IEC 61508—Design of electrical/electronic/programmable electronic safety-related systems*. Computing and Control Engineering Journal. Vol. 11(1). (pp. 6-12).

[Bro10] Brooks, F.P. (2010). *The Design of Design: Essays from a Computer Scientist*. Addison Wesley.

[Bun08] Bunge, M. (2008). *Causality and Modern Science*. Transaction Publishers.

[Bur09] Burns, A. & A. Wellings. (2009). *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*. Addison-Wesley.

[But04] Buttazzo, G. (2004). *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Springer Verlag.

[CAN90] CAN. (1992). *Controller Area Network CAN, an In-Vehicle Serial Communication Protocol*. SAE Handbook 1992. SAE Press. (pp. 20.341-20.355).

[Car08] Cardenas, A., A., S. Amin, & S. Shastry. (2008). *Research Challenges for the Security of Control Systems*. Proc. of the Workshop on *Hot Topics in Security*. Usenix Association. URL: http://portal.acm.org/citation.cfm?id=1496671.1496677

[Cha09] Chandola, V., A. Banerjee & V. Kumar. (2009). *Anomaly Detection: A Survey*. ACM Computing Surveys. Vol. 41(3). (pp. 15.1-15.58.)

[Che87] Cheng, S.C. (1987). *Scheduling Algorithms for Hard Real-Time Systems–A Brief Survey*. In: *Hard Real-Time Systems*. IEEE Press.

[Chu08] Chu, Y., Burns, A. (2008). *Flexible Hard Real-Time Scheduling for Deliberative AI Systems*. Real-Time Systems Journal. Vol. 40(3). (pp. 241-263).

[Cla03] Clark, E., et al. (2003). *Counterexample-Guided Abstraction Refinement for Symbolic Model Checking*. Journal of the ACM. Vol. 50(5). (pp. 752-794).

[Cla88] Clark, D. (1988). *The Design Philosophy of the DARPA Internet Protocols*. Computer Communication Review. Vol. 18(4). (pp. 106-114).

[Con02] Constantinescu, C. (2002). *Impact of Deep Submicron Technology on Dependability of VLSI Circuits*. Proc. of DSN 2002. IEEE Press. (pp. 205-209).

[Cri89] Cristian, F. (1989). *Probabilistic Clock Synchronization*. Distributed Computing. Vol. 3. Springer Verlag. (pp. 146-185).

[Cum10] Cumming, D.M. (2010). *Haven't found that software glitch, Toyota? Keep trying*. Los Angeles Times. March 11, 2010.

[Dav79] Davies, C.T. (1979). *Data Processing Integrity*. In: *Computing Systems Reliability*. Cambridge University Press. (pp. 288-354).

[Dea02] Deavours, D., et al. (2002). *The Mobius framework and its implementation*. IEEE Trans. on Software Engineering, Vol. 28(10). (pp. 1-15).

[Deg95] Degani, A., Shafto, M. & Kirlik, A. (1995). *Mode Usage in Automated Cockpits: some Initial Observations*. Proc. of IFAC 1995. IFAC Press. (pp. 1-13).

[Dri03] Driscoll, K. et. a. (2003). *Byzantine Fault-Tolerance: From Theory to Reality*. Proc. of SAFECOMP 2003. LNCS 2788. Springer Verlag. (pp. 235-248).

[Dys98] Dyson, G.B. (1998). *Darwin among the Machines—the Evolution of Global Intelligence*. Basic Books.

[Edm00] Edmonds, B., (2000). *Complexity and Scientific Modeling*. In: *Foundations of Science*. Springer Verlag. (pp. 379-390).

[Eid06] Eidson, J. (2006). *Measurement, Control and Communication Using IEEE 1588*. Springer Verlag.

[Eps08] Epstein, S. (2008). *Intuition from the Perspective of Cognitive Experiential Self-Theory* . In: *Intuition in Judgment and Decision Making*. Lawrence Erlbaum New York. (pp. 23-38).

[Fei06] Feiler, P., D. Gluch, & J. Hudak. (2006). *The Architecture Analysis and Design Language (AADL): An Introduction*. Report CMU-SEI 2006-TN-011. Software Engineering Institute.

[Fel04] Feltovich, P.J., et al. (2004). *Keeping it too Simple: How the Reductive Tendency Effects Cognitive Engineering*. IEEE Intelligent Systems. May/June 2004, IEEE Press (pp. 90-94).

[Fel04a] Feldhofer, M., S. Dominikus, & J. Wokerstorfer. (2004). *Strong Authentication for RFID Systems Using the AES Algorithms*. LCNS 3156. Springer Verlag. (pp. 357-370).

[Fin03] Finkenzeller, K. (2003). *RFID Handbook*. John Wiley and Sons.

[Fis06] Fisher, D.A (2006). *An Emergent Perspective on the Operation of System-of-Systems*. Carnegie Mellon Software Engineering Institute. CMU/SEI-2006-TR-003. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA449020

[Foh94] Fohler, G. (1994). *Flexibility in Statically Scheduled Hard Real-Time Systems*. PhD Thesis. Institut für Technische Informatik. Technical University of Vienna.

[Fra01] Frank, D., et al. (2001). *Device Scaling Limits of Si MOSFETs and Their Application Dependencies*. Proc. of the IEEE. Vol. 89(3). (pp. 259-288).

[Gad10] Gadh, R., et al. (2010). RFID: *A unique Radio Innovation for the 21st Century*. Proc. of the IEEE. Vol. 98(2). (pp. 1541-1680).

[Gaj09] Gajski, D.D., et al. (2009). *Embedded System Design*. Springer Verlag.

[Gar75] Garey, M.R. & D.S. Johnson. (1975). *Complexity Results for Multiprocessor Scheduling under Resource Constraints*. SIAM Journal of Computing. Vol. 4(4). (pp. 397-411).

[Gau05] Gaudel, M.C. (2005). *Formal Methods and Testing: Hypotheses, and Correctness Approximations*. Proc. of Formal Methods 2005. LNCS 3582. Springer Verlag.

[Gra85] Gray, J. (1985). *Why do Computers Stop and What can be done about it?* Tandem Technical Report TR85.7. Cupertino, CA.

[Hal05] Halford, G.S., et al., How Many Variables Can Humans Process? Psychological Science, 2005. 16(1): pp. 70-76.

[Hal92] Halbwachs, N., (1992). *Synchronous Programming of Reactive Systems*. Springer Verlag.

[Hal96] Halford, G.S., W.H. Wilson, & S. Phillips. (1996). *Abstraction, Nature, Costs, and Benefits*. Department of Psychology, University of Queensland, 4072 Australia.

[Hay90] Hayakawa, S.I. (1990). *Language in Thought and Action*. Harvest Original, San Diego.

[Hei00] Heinzelman, W.R., A. Chandrakasan, & H. Balakrishan. (2000). *Energy-efficient Communication Protocol for Wireless Microsensor Networks*. Proc. of the 33rd Hawaii International Conference on System Science. IEEE Press. (pp. 3722-3725).

[Hen03] Henzinger, T., B. Horowitz, & C.M. Kirsch, Giotto: A Time-Triggered Language for Embedded Programming. Proc. of the IEEE, 2003. 91(1): pp. 84-99.

[Her09] Herault, L. (2009). *Holistic Approach for Future Energy-Efficient Cellular Networks*. Proc. of the Second Japan-EU Symposium on the Future Internet. European Communities Brussels. (pp. 212-220).

[Hme04] Hmelo-Silver, C.E. & M.G. Pfeffer. (2004*). Comparing Expert and Novice Understanding of a Complex System from the Perspective of Structures, Behaviors, and Functions*. Cognitive Science, Elsevier, Vol. 28. (pp. 127-138).

[Hoe10] Hoefer, C. (2010). *Causal Determinism.* In: *Stanford Encyclopedia of Philosophy*. (pp. 1-24). URL: http://plato.stanford.edu/entries/determinism-causal/

[Hop78] Hopkins, A.L., T.B. Smith, & J.H. Lala. (1978). *FTMP: A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft Control*. Proc. IEEE. Vol 66(10). (pp. 1221-1239).

[Hue08] Huebscher, M.C. & J.A. McCann. (2008). *A Survey of Autonomic Computing–Degrees, Models and Applications*. ACM Computing Surveys. Vol. 40(3). (pp. 7.1-7.28).

[Int09] Intel. (2009). *Teraflop Research Chip*. URL: http://techresearch.intel.com/articles/Tera-Scale/1449.htm.

[ITR09] ITRS Roadmap. (2009). *International Technology Roadmap for Semiconductors, 2009 Edition. Executive Summary*. Semiconductor Industry Association.

[Jac07] Jackson, D., M. Thomas, & L.I. Millet. (2007). *Software for Dependable Systems: Sufficient Evidence*? National Academic Press. Washington.

[Jer77] Jerri, A.J.)1977). *The Shannon Sampling Theorem—Its Various Extensions and Applications: A Tutorial Review*. Proc. of the IEEE. Vol. 65(11). (pp. 1565-1596).

[Joh92] Johnson, S. C., & Butler, R. W. (1992). *Design for Validation*. IEEE Aerospace and Electronic Systems Magazine. Vol. 7(1). (pp. 38-43).

[Jon78] Jones, J., C. (1978). *Design Methods, Seeds of Human Futures*. John Wiley.

[Jon97] Jones, M. (1997). *What really happened on Mars Rover Pathfinder*. URL: http://catless.ncl.ac.uk/Risks/19.49.html#subj1.

[Jue05] Juels, A. & S.A. Weis. (2005). *Authenticating Pervasive Devices with Human Protocols*. In: Proc. of CRYPTO 2005. Springer Verlag. (pp. 293-308).

[Jur04] Juristo, N., A.M. Moreno, & S. Vegas, (2004). *Reviewing 25 Years of Testing Technique Experiments*. In: *Empirical Software Engineering*, Vol. 9. Springer Verlag. (pp. 7-44)

[Kan95] Kantz, H. & C. Koza. (1995). *The ELECTRA Railway Signaling-System: Field Experience with an Actively Replicated System with Diversity*. Proc. FTCS 25. (pp. 453-458).

[Kar95] Karlson, J. et al. (1995). *Integration and Comparison of Three Physical Fault-Injection Experiments*. In: *Predictably Dependable Computing Systems*. Springer Verlag.

[Kea07] Keating, M., et al. (2007). *Low Power Methodology Manual for Chip Design*. Springer Verlag.

[Kim94] Kim, K.H. & H. Kopetz. (1994). *A Real-Time Object Model RTO.k and an Experimental Investigation of its Potential*. Proc. COMPSAC 94. IEEE Press.

[Kni86] Knight, J.C. & N.G. Leveson. (1986). *An Experimental Evaluation of the Assumption of Independence in Multiversion Programming*. IEEE Trans. Software Engineering. Vol. SE-12(1). (pp. 96-109).

[Koo04] Koopman, P. (2004). *Embedded System Security*. IEEE Computer, (July 2004). (pp. 95-97.)

[Kop03] Kopetz, H, & Suri, N. (2003*). Compositional Design of RT Systems: A conceptual Basis for the Specification of Linking Interfaces*. Proc. of 6th ISORC. IEEE Press. (pp. 51-59).

[Kop03a] Kopetz, H. & G. Bauer. (2003). *The Time-Triggered Architecture*. Proc. of the IEEE, Vol. 91(1). (pp. 112-126).

[Kop04] Kopetz, H., A. Ademai, & A. Hanzlik.(2004). *Integration of Internal and External Clock Synchronization by the Combination of Clock State and Clock Rate Correction in Fault Tolerant Distributed Systems.* Proc. of the RTSS 2004. IEEE Press. (pp. 415-425).

[Kop06] Kopetz, H. (2006). *Pulsed Data Streams*. In: *From Model-Driven Design to Resource Management for Distributed Embedded Systems*. IFIP Series 225/2006. Springer Verlag. (pp. 105-114).

[Kop07] Kopetz, H., et al. (2007). *Periodic Finite-State Machines.* Proc. of ISORC 2007. IEEE Press. (pp. 10-20).

[Kop08] Kopetz, H. (2008). *The Rationale for Time-triggered Ethernet*. RTSS 2008. IEEE Press. (pp. 3-11).

[Kop08a] Kopetz, H. (2008). *The Complexity Challenge in Embedded System Design*. Proc. of ISORC 2008. IEEE Press. (pp. 3-12).

[Kop09] Kopetz, H. (2009). *Temporal Uncertainties in Cyber-Physical Systems*. Institute für Technische Informatik, TU Vienna, Report 1/2009. Vienna.

[Kop10] Kopetz, H. (2010). *Energy-Savings Mechanism in the Time-Triggered Architecture.* Proc. of the 13th ISORC. IEEE Press. (pp. 28-33).

[Kop85] Kopetz, H. & W. Merker. (1985). *The Architecture of MARS*. Proc. of FTCS-15. IEEE Press. (pp. 274-279).

[Kop87] Kopetz, H. & W. Ochsenreiter. (1987). *Clock Synchronization in Distributed Real-Time Systems*. IEEE Trans. Computers. Vol. 36(8). (pp. 933-940).

[Kop89] Kopetz, H. et al. (1989). *Distributed Fault Tolerant Real-Time Systems: The MARS Approach*. IEEE Micro. Vol. 9(1). (pp. 25-40).

[Kop90] Kopetz, H. & K. Kim. (1990). *Temporal Uncertainties in Interactions among Real-Time Objects.* Proc. 9th IEEE Symp. on Reliable Distributed Systems. IEEE Press. (pp. 165-174).

[Kop91] Kopetz, H., G. Grünsteidl, & J. Reisinger. (1991). *Fault-Tolerant Membership Service in a Synchronous Distributed Real-Time System.* In: *Dependable Computing for Critical Applications.* Springer-Verlag. (pp. 411-429).

[Kop92] Kopetz, H. (1992). *Sparse Time versus Dense Time in Distributed Real-Time Systems*. Proc. 14th Int. Conf. on Distributed Computing Systems. IEEE Press. (pp. 460-467).

[Kop93] Kopetz, H. & G. Gruensteidl. (1993). *TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems.* Proc. FTCS-23. IEEE Press. (pp. 524-532)

[Kop93a] Kopetz, H. & J. Reisinger. (1993). *The Non-Blocking Write Protocol NBW: A Solution to a Real-Time Synchronisation Problem*. Proc. of RTSS 1993. IEEE Press. (pp. 131-137).

[Kop95] Kopetz, H. (1995). *The Time-Triggered Approach to Real-Time System Design*. In: *Predictably Dependable Computing Systems*. Brian Randell, Editor. Springer Verlag. (pp. 53-66).

[Kop98] Kopetz, H. (1998). *The Time-triggered Model of Computation*. Proc. of RTSS 1998. IEEE Press. (pp. 168-176).

[Kop99] Kopetz, H. (1999). *Elementary versus Composite Interfaces in Distributed Real-Time Systems.* Proc. of ISADS 99. IEEE Press. (pp. 26-34).

[Kor10] Kortuem, G., et al. (2010). *Smart Objects as the Building Block for the Internet of Things*. IEEE Internet Computing, Jan 2010). (pp. 44-50).

[Kum10] Kumar, K. & Y.H. Lu (2020). *Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?* IEEE Computer, April 2010. (pp. 51-56).

[Lal94] Lala, J.H. & R.E. Harper. (1994). *Architectural Principles for Safety-Critical Real-Time Applications.* Proc. of the IEEE. Vol. 82(1). (pp. 25-40).

[Lam74] Lamport, L. (1994). *A New Solution of Dijkstra's Concurrent Programming Problem*. Comm. ACM. Vol. 8(7). (pp. 453-455).

[Lam78] Lamport, L.(1978). *Time, Clocks, and the Ordering of Events*. Comm. ACM. Vol. 21 (7). (pp. 558-565).

[Lam82] Lamport, L., Shostak, R., Pease, M. (1982). The Byzantine Generals Problem. Comm. ACM TOPLAS. Vol. 4(3). (pp. 382-401).

[Lam85] Lamport, L. & P.M. Melliar Smith. (1985). *Synchronizing Clocks in the Presence of Faults*. Journal of the ACM. Vol. 32(1). (pp. 52-58).

[Lan09] Langley, P., J.E. Laird, & S. Rogers. (2009). *Cognitive Architectures: Research Issues and Challenges*. Cognitive System Research. Vol. 10(2). (pp. 141-160).

[Lan81] Landwehr, C.E. (1981). *Formal Models for Computer Security*. ACM Computing Suverys. Vol. 13(3). (pp. 248-278).

[Lan97] Landwehr, C.E. & D.M. Goldschlag. (1997). *Security Issues in Networks with Internet Access*. Proc. of the IEEE. Vol. 85(12). (pp. 2034-2051).

[Lau06] Lauwereins, R. (2006). *Multi-core Platforms are a Reality. . .but where is the Software Support?* Visual Presentation. IMEC. URL: http://www.mpsoc-forum.org/2006/slides/Lauwereins.pdf

[Lee02] Lee, E.A. (2002). *Embedded Software*. Advances in Computers. Vol. 56. Academic Press

[Lee10] Lee, E.A. & Seshia, S. A. (2010). *Introduction to Embedded Systems – A Cyber-Physical Systems Approach*. http://LeeSeshia.org, 2010.

[Lee90] Lee, P.A. & Anderson, T. (1990). *Fault Tolerance: Principles and Practice*. Springer Verlag.

[Leh85] Lehmann M.M. & Belady, L. (1985). *Program Evolution: Processes of Software Change*. Academic Press.

[Lev08] Leverich, J., et al. (2008). *Comparative Evaluation of Memory Models of Chip Multiprocessors*. ACM Trans. on Architecture and Code Optimization, 2008. Vol. 5 (3). (pp. 12.1-12.30).

[Lev95] Leveson, N.G. (1995). *Safeware: System Safety and Computers*. Addison Wesley Company. Reading, Mass.

[Lie10] Lienert, D., Kriso, S. (2010). *Assessing Criticality in Automotive Systems*. IEEE Computer. Vol. 43(5). (p. 30).

[Lit93] Littlewood, B. & L. Strigini, (1993). *Validation of ultra-high dependability for software-based systems*. Comm. ACM. Vol. 36(11). (pp. 69-80).

[Liu00] Liu, J.W.S. (2000). *Real-Time Systems*. Prentice Hall.

[Liu73] Liu, C.L. & J.W. Layland. (1973). *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment*. Journal of the ACM. Vol. 20(1). (pp. 46-61).

[Lun84] Lundelius, L. & N. Lynch. (1984). *An Upper and Lower Bound for Clock Synchronization*. Information and Control. Vol. 62. (pp. 199-204).

[Lv09] Lv, M. et al. (2009). A Survey of WCET Analysis of Real-Time Operating Systems. URL: http://www.neu-rtes.org/publications/lv_ICESS09.pdf

[Mai98] Maier, M.W. (1998). *Architecting Principles for System of Systems*. Systems Engineering.Vol. 1(4). (pp. 267-284).

[Mar10] Marwedel, P. (2010). *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems*. Springer Verlag.

[Mar91] MARS. (1991). *The Mars Video*. TU Vienna, URL: http://pan.vmars.tuwien.ac.at/mars/video

[Mar99] Martin, T.L. & D.P. Siewiorek. (1999). *The Impact of Battery Capacity and Memory Bandwidth on CPU Speed Setting: A Case Study*. Proc. of ISLPED99. IEEE Press. (pp. 200-205).

[McC09] McCabe, M. et al. (2009). *Avionics Architecture Interface Considerations between Constellation Vehicles*. Proc. DASC'09. IEEE Press. (pp. 1.E.2.1-1.E.2,10).

[Mes04] Mesarovic, M.D., S.N. Screenath, & J.D. Keene. (2004) *Search for Organizing Principles: Understanding in Systems Biology*. Systems Biology on line, Vol. 1(1). (pp. 19-27).

[Mes89] Mesarovic, M.D. (1989). *Abstract System Theory*. Lecture Notes in Control and Information Science. Vol. 116. Springer Verlag.

[Met76] Metclafe, R.M., Ethernet. (1976). *Distributed Packet Switching for Local Computer Networks*. Comm. of the ACM. (pp. 395-404).

[Mil04] Miller, D. (2004). *AFDX Determinism*. Visual Presentation at ARINC General Session, October 27, 2004. Rockwell Collins.

[Mil56] Miller, G.A. (1956). *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*. The Psychological Review. Vol. 63. (pp. 81-97).

[Mil91] Mills, D.L. (1991). *Internet Time Synchronization: The Network Time Protocol*. IEEE Trans. on Comm. Vol. 39(10). (pp. 1482-1493).

[Mog06] Mogul, C. (2006). *Emergent (Mis)behavior vs. Complex Software Systems*. Proc. of EuroSys 2006. ACM Press.

[Mok93] Mok, A. (1993). *Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment*. PhD Thesis. Massachusetts Institute of Technology.

[Mor07] Morin, E. (2007). *Restricted Complexity, General Complexity*. World Scientific Publishing Corporation.

[NAS99] NASA, (1999). *Mars Climate Orbiter Mishap Investigation Report*. Washington, DC. ftp://ftp.hq.nasa.gov/pub/pao/reports/2000/MCO_MIB_Report.pdf

[Neu56] Neumann, J. (1956). *Probabilistic Logic and the Synthesis of Reliable Organisms from Unreliable Components. In: Automata Studies, Annals of Mathematics Studies* No. 34, C.E. Shannon & M. J, Editors. Princeton Univ. Press. (pp. 43-98).

[Neu94] Neuman, B.C. & T. Ts'o. (1994). *Kerberos–An Authentication Service for Computer Networks*. IEEE Communication Magazine. Vol. 32(9). (pp. 33-38).

[Neu95] Neumann, P.G. (1995). *Computer Related Risks*. Addison Wesley—ACM Press.

[Neu96] Neumann, P.G. (1996). *Risks to the Public in Computers and Related Systems*. Software Engineering Notes, Vol. 21(5). ACM Press. p. 18.

[Obm09] Obermaisser, R. & H. Kopetz. (2009). *GENSYS–An ARTEMIS Cross-Domain Reference Architecture for Embedded Systems*. Südwestdeutscher Verlag für Hochschulschriften (SVH).

[OMG08] OMG, MARTE. (2008). *Modeling and Analysis of Real-time and Embedded Systems,* Object Management Group.

[Pap02] Pappu, R., et al. (2002). *Physical One-Way Functions*. Science. Vol. 297. (pp. 2026-2030).

[Par97] Parunak, H.v.D. et. al. [1997]. *Managing Emergent Behavior in Distributed Control Systems*. Proc. of ISA Tech 97. (pp. 1-8).

[Pau08] Paukovits, C. & H. Kopetz. (2008). *Concepts of Switching in the Time-Triggered Network-on-Chip*. 14th IEEE Conference on Embedded and Real-Time Computing Systems and Applications. IEEE Press. (pp. 120-129).

[Pau98] Pauli, B., A. Meyna, & P. Heitmann. (1998). *Reliability of Electronic Components and Control Units in Motor Vehicle Applications*. Verein Deutscher Ingenieure (VDI). (pp. 1009-1024).

[Pea80] Pease, M., R. Shostak, & L. Lamport, Reaching Agreement in the Presence of Faults. Journal of the ACM, 1980. 27(2): pp. 228-234.

[Ped06] Pedram, M. & S. Nazarian. (2006). *Thermal Modeling, Analysis and Management in VLSI Circuits: Principles and Methods*. Proc. of the IEEE. Vol. 94(8). (pp. 1487-1501).

[Per10] Perez, J.M. (2010). Executable Time-Triggered Model (E-TTM) for the Development of Safety-Critical Embedded Systems. PhD. Thesis, Institut für Technische Informatik, TU Wien, Austria. (pp. 1-168).

[Pet79] Peters, L. (1979). Softwa*re Design: Current Methods and Techniques* .In: *Infotech State of the Art Report on Structured Software Development*. Infotech International. London.

[Pet96] Peterson, I. (1996). *Comment on Time on Jan 1, 1996*. Software Engineering Notes, Vol. 19(3). (p. 16).

[Pol07] Polleti, F., et al. (2007). *Energy-Efficient Multiprocessor Systems-on-Chip for Embedded Computing: Exploring Programming Models and Their Architectural Support*. Proc. of the IEEE, 2007. Vol. 56(5). (pp. 606-620).

[Pol95] Poledna, S. (1995). *Fault-Tolerant Real-Time Systems, The Problem of Replica Determinism*. Springer Verlag.

[Pol95b] Poledna, S. (1995). *Tolerating Sensor Timing Faults in Highly Responsive Hard Real-Time Systems*. IEEE Trans. on Computers. Vol. 44(2). (pp. 181-191).

[Pop68] Popper, K.R. (1968). *The Logic of Scientific Discovery*. Hutchinson, London.

[Pow91] Powell, D. (1991). *Delta-4, A Generic Architecture for Dependable Distributed Computing*. Springer Verlag. (pp. 1-484).

[Pow95] Powell, D. (1995). *Failure Mode Assumptions and Assumption Coverage* In: B. Randell, J. C. Laprie, H. Kopetz, & B. Littlewood (Ed.), *Predictably Dependable Computing Systems*. Springer Verlag. Berlin. (pp. 123-140).

[Pus89] Puschner, P. & C. Koza. (1989). *Calculating the Maximum Execution Time of Real-Time Programs*. Real-Time Systems. Springer Verlag. Vol: 1(2). (pp. 159-176).

[Ran75] Randell, B. (1975). *System Structure for Software Fault Tolerance*. IEEE Trans. on Software Engineering, Vol. SE-1(2). (pp. 220-232).

[Ray10] Ray, K. (2010). *Introduction to Service-Oriented Architectures*. URL: http://anengineersperspective.com/wp-content/uploads/2010/03/Introduction-to-SOA.pdf

[Rec02] Rechtin, E. & M.W. Maier. (2002). *The Art of Systems Architecting*. CRC Press.

[Rec91] Rechtin, E. (1991). *Systems Architecting, Creating and Building Complex Systems*. Prentice Hall.

[Rei10] Reisberg, D. (2010). *Cognition*. W.W. Norton, New York, London.

[Rei57] Reichenbach, H. (1957). *The Philosophy of Space and Time*. Dover, New York.

[Riv78] Rivest, R.L., A. Shamir, & L. Adleman. (1978). *A Method for Obtaining Signatures and Public-Key Cryptosystems*. Comm. of the ACM. Vol. 21(2). (pp. 120-126).

[Rom07] Roman, R., C. Alcarez, & J. Lopez. (2007). *A Survey of Cryptographic Primitives and Implementations for Hardware-Constrained Sensor Network Nodes*. Mobile Network Applications. Vol. 12. (pp. 231-244). Springer Verlag.

[Rum08] Rumpler, B., (2008) Design Comprehension of Embedded Rea-Time Systems. PhD Thesis. Institut für Technische Informatik. TU Wien.

[Rus99] Rushby, J. (1999). *Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms*. IEEE Trans. Software Engineering. Vol. 25(5). (pp. 651-660).

[Rus03] Rushby, J. (2003). *A Comparison of Bus Architectures for Safety Critical Embedded Systems*. Report NASA/CR–2003–212161.

[Rus93] Rushby, J. (1993). *Formal Methods and the Certification of Critical Systems* (Research Report No. SRI-CSL-93-07). Computer Science Lab, SRI, Menlo Park, Cal.

[SAE95] SAE. (1995). *Class C Application Requirements, Survey of Known Protocols, J20056*. In: *SAE Handbook*. SAE Press. (pp. 23.437-23.461).

[Sal10] Salfner, F., Lenk, M., and Malek, M. (2010). *A Survey of Online Failure Prediction Methods*. ACM Computing Surveys. Vol. 42(3). (pp. 10.10-10.42).

[Sal84] Saltzer, J., Reed, D. P., & Clark, D. D. (1984). *End-to-End Arguments in System Design*. ACM Trans. on Computer Systems. Vol. 2(4). (pp. 277-288).

[Sch88] Schwabl, W.(1988). *The Effect of Random and Systematic Errors on Clock Synchronization in Distributed Systems*. PhD Thesis. Technical University of Vienna, Institut für Technische Informatik.

[Sel08] Selberg, S.A. & Austin, M.A. (2008]. *Towards an Evolutionary System of Systems Architecture*. Institute for Systems Research.URL: http://ajcisr.eng.umd.edu/~austin/reports.d/INCOSE2008-Paper378.pdf

[Ses08] Sessions, R. (2008). *Simple Architectures for Complex Enterprises*. Published by Microsoft Press.

[Sev81] Sevcik, F. (1981). *Current and Future Concepts in FMEA*. Proc. of the Annual Reliability and Maintainability Symposium. Philadelphia. IEEE Press. (pp. 414-421).

[Sha04] Sha, L. et al. (2004). *Real-Time Scheduling Theory: A Historical Perspective*. Real-Time Systems Journal. Vol. 28(3/4). Springer Verlag. (pp. 101-155).

[Sha89] Shaw, A.C. (1989). *Reasoning About Time in Higher-Level Language Software*. IEEE Trans. on Software Engineering. Vol. SE-15. (pp. 875-889).

[Sha90] Sha, L., R. Rajkumar, & J.P. Lehoczky. (1990). *Priority Inheritence Protocols: An Approach to Real-Time Synchronization*. IEEE Trans. on Computers. Vol. 39(9). (pp. 1175-1185).

[Sha94] Sha, L., R. Rajkumar, and S.S. Sathaye. (1994). *Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems*. Proc. of the IEEE. Vol. 82(1). (pp. 68-82).

[Sie00] Siegel, J. (2000). *CORBA 3–Fundamentals and Programming*. OMG Press. John Wiley.

[Sim81] Simon, H.A. (1981). *Science of the Artificial*. MIT Press.

[Smi97] Smith, J.S.S. (1997). *Application Specific Integrated Circuits*. Addision Wesley.

[Soe01] Soeleman, H., K. Roy, & B.C. Paul. (2001). *Robust Subtreshold Logic for Ultra-Low Power Operation*. IEEE Trans. on VLSI Systems. Vol. 9(1). (pp. 90-99).

[Spr89] Sprunt, B., L. Sha, & J. Lehoczky. (1989). *Aperiodic Task Scheduling for Hard Real-Time Systems*. Real-Time Systems. Vo. 1(1). (pp. 27-60).

[Sta03] Stamatis, D.H. (2003). *Failure Mode and Effect Analysis: FMEA from Theory to Execution*. ASQ Quality Press.

[Sta08] Stallings, W. (2008). *Operating Systems: Internals and Design Principles*. Prentice Hall.

[Szy99] Szyperski, C. (1999). *Component Software—Beyond Object-Oriented Programming*. Addision Wesley.

[Tai03] Taiani, F., J.C. Fabre, & M.O. Killijian. (2003). *Towards Implementing Multi-Layer Reflection for Fault-Tolerance*. Proc. of the DSN 2003. IEEE Press. (pp. 435-444).

[Tas03] Task Force. (2004). *Final Report on the August 14, 2003 Blackout in the United States and Canada*. US Department of Energy.

[Ter11] Terzija, V. et. al. (2011). *Wide-Area Monitoring, Protection, and Control of Future Electric Power Networks*. Proc. of the IEEE. Vol. 99(1). (pp. 80-93).

[Tel09] Telecom Japan. (2009). *Cyber-Clean Center (CCC) Project for Anti-Bot Countermeasures in Japan*. Proc. of the Second Japan-EU Symposium on the Future Internet. European Communities Brussels. (pp. 212-220).

[Tin95] Tindell, K. (1995). Analy*sis of Hard Real-Time Communications*. Real-Time Systems. Vol. 9(2). (pp. 147-171).

[Tra88] Traverse, P. (1988). *AIRBUS and ATR System Architecture and Specification*. In: *Software Diversity in Computerized Control Systems*. Springer-Verlag.

[Ver09] Vermesan, O. et al. (2009). *Internet of Things—Strategic Research Roadmap*. European Commission-Information Society and Media DG. Brussels.

[Ver94] Verissimo, P. (1994). *Ordering and Timeliness Requirements of Dependable Real-Time Programs*. Real-Time Systems. Vol. 7(3). (pp. 105-128).

[Vig10] Vigras, W.J. (2010). *Calculation of Semiconductor Failure Data*. URL: http://rel.intersil.com/docs/rel/calculation_of_semiconductor_failure_rates.pdf

[Vig62] Vigotsky, L.S. (1962). *Thought and Language*. MIT Press, Boston, Mass.

[Wen78] Wensley, J.H., et al. (1978). *SIFT: The Design and Analysis of a Fault-Tolerant Computer for Aircraft Control*. Proc. IEEE. Vol. 66(10). (pp. 1240-1255).

[Wik10] Washington's Axe. (2010). Wikipedia, URL: http://en.wikipedia.org/wiki/George_-Washington%27s_axe#George_Washington.27s_axe

[Wil08] Wilhelm, R. et al. (2008). *The Worst-Case Execution Time Problem—Overview of Methods and Survey of Tools*. ACM Trans. on Embedded Computer Systems, Vol. 7(3). (pp. 1-53).

[Win01] Winfree, A.T. (2001). *The Geometry of Biological Time*. Springer Verlag.

[Wit90] Withrow, G.J. (1990). *The Natural Philosophy of Time*. Oxford Science Publications. Clarendon Press, Oxford.

[Xin08] Xing, L. Amari, S.V. (2008). *Handbook of Performability Engineering*. Springer Verlag.

[Xu91] Xu, J., & Parnas, D. (1990). *Scheduling Processes with Release Times, Deadlines, Precedence, and Exclusion Relations*. IEEE Trans. on Software Engineering. Vol. 16 (3). (pp. 360-369).

[Zil96] Zilberstein, S. (1996). *Using Anytime Algorithms in Intelligent Systems*. AI Magazine. Vol. 16(3). (pp. 73-83).

Index

A
AADL. *See* Architecture analysis and design language
Abstraction, 30, 32–46, 48, 49, 58, 76, 86, 87, 92, 94, 103, 105, 108, 112, 130, 136, 155, 203, 231, 232, 244, 261, 266–270, 301, 302
  ladder, 41, 44
Accuracy, 15, 23, 25, 55–56, 73, 76, 87, 115–122, 128, 149, 159, 168, 221, 226, 329
  interval, 4, 117–120, 124, 131, 227
Acknowledgment, 91, 169, 170, 175, 179
Action delay, 123–124, 131, 168, 182, 187
Activation energy, 199
Advanced encryption standard (AES), 145, 147, 148
Adversary argument, 242–243, 256
AES. *See* Advanced encryption standard
AFDX. *See* Avionics full duplex switched ethernet
Agreed data, 5, 128, 233
Agreement protocols, 63–65, 76, 126–128, 157, 158, 182, 215, 231–234
Alarm monitoring, 5, 52, 82, 83, 110, 122, 123, 171, 177, 187, 222, 311
ALARP. *See* As low as reasonably practical
Alternative scheduling strategies, 239, 255–256
Analog input/output, 226–227
Analysis, 5, 16, 31–33, 36, 38, 41, 42, 46, 47, 54, 61, 86, 143, 151, 164, 205, 208, 221, 236, 244–247, 252, 253, 260, 263–266, 268, 269, 274–280, 282, 286, 288, 292, 298, 300–302
Analytic rational, 29–31, 46, 47
Anomaly detection, 144, 150–153, 161, 164, 284–285, 295–297, 331
Anytime algorithm, 208, 246–247, 257
Aperiodic task, 242
API. *See* Application programming interface
Application programming interface (API), 217, 219–221, 262, 268
Architectural style, 40, 48, 82, 96–98, 106, 108, 208, 318, 325, 328–332, 337
Architecture, 11, 38, 69–70, 101–108, 110, 121, 140, 142, 145–146, 159, 164, 172, 187, 188, 203–207, 209, 230, 247, 262, 263, 265, 272, 274, 278, 279, 283, 301–303, 312, 315, 325–339
Architecture analysis and design, 92, 260, 268
Architecture analysis and design language (AADL), 269
Architecture design languages, 268–269
ARINC 629 protocol, 130, 167, 181, 187
As low as reasonably practical (ALARP), 274, 279
Assertion, 100, 150, 152, 296
Assumption coverage, 16
Atomic, 38, 53, 56, 73, 81, 85–88, 90, 104, 107, 113, 130–131, 162, 169, 225, 313, 321
Attack strategy, 143, 281
Attribute, 10, 13, 39, 48, 98, 100, 112, 116, 269, 317, 319
Audio/video bridging (AVB), 183
Audio video bus, 182–183
Autonomic components, 314–315
Availability (A), 12–13, 15, 16, 20, 26, 38, 47, 66, 107, 141, 142, 161, 164, 168, 176, 208, 221, 276, 285, 308, 319, 327–329, 337
AVB. *See* Audio/video bridging
Avionics full duplex switched ethernet (AFDX), 127, 167, 181, 182

B
Babbling idiot, 103, 169, 180, 185, 187, 304
Back-pressure flow control, 89, 175, 177, 179, 180
Bandwidth, 21, 22, 66, 127, 130, 132, 167, 171, 174, 175, 178, 180–182, 205
Basic level concepts, 36–38, 41
Basic message transport service (BMTS), 38–39, 169, 171–174, 178–179, 187, 217, 293, 299, 330–331
Batteries, 19, 191, 193, 197, 202–204, 208–212, 262, 268, 314, 317, 320, 321
Behavior, 2, 3, 5, 18, 30, 34, 35, 41–47, 49, 52, 68, 71, 80, 83, 84, 86–88, 99, 103, 105–108, 121, 126, 127, 130, 131, 136, 139–141, 150–152, 155, 157, 171, 172, 240, 256, 257, 262, 263, 268, 292, 295, 298, 299, 302, 315, 331
Benign failure, 11, 12, 26, 140, 176
Best-effort, 13, 16, 175, 176, 179, 180, 208, 255, 257
Bit length of a channel, 174, 175, 188
Blocking synchronization statement, 220
BMTS. *See* Basic message transport service
Bohrbug, 138
Botnet attack, 145
Bus guardian, 185, 304
Byzantine, 67, 68, 70–72, 77, 140, 141, 155, 158, 233, 271

C
Causality, 5, 30, 31, 35, 46–47, 52–53, 76, 80, 125, 268–269
Causal order, 52–53, 76
Celestial mechanics, 41
Central master synchronization, 68–69
Certification, 11, 216, 272, 276–277, 279–281, 288, 299, 302, 333
Checkpoint recovery, 14, 15, 87–88
Cipher, 145–148, 335
Clean failure, 19–20, 139–140
Clock, 2, 53, 80, 116, 148, 168, 195, 219, 248, 269, 302, 317, 326
  drift, 54–56, 66, 67, 70, 72–73
Closed-loop control, 8, 20
Cloud computing, 208, 314
Cluster, 2–4, 46, 64, 73–75, 77, 81–82, 92, 94–98, 103, 104, 108, 159, 162–164, 185, 218–219, 265, 269–271, 284, 315, 330, 332, 334–338
CNI. *See* Communication network interface
Cognitive complexity, 33–35, 48, 309, 328
Communication architecture, 103, 121, 159, 172, 313–314, 337, 338
Communication network interface (CNI), 2, 81, 104
Communication system, 24, 46, 53, 81–82, 84, 89, 91, 93, 96, 99, 103, 120, 123–125, 127, 128, 141, 157, 159, 168, 169, 171, 172, 174–181, 183, 186, 187, 224–225, 255, 270, 278, 298, 329, 331, 334
Compiler analysis, 244–245
Complex systems, 34, 44–45, 104–106, 204, 251, 255, 265, 274
Complex task (C-task), 220–221, 236, 241, 243, 246, 251
Component, 21, 40, 80, 119, 136, 168, 203, 216, 248, 260, 294, 308, 327
  cycle, 141, 160, 162–164
  restart, 94, 155, 161, 163–164, 205–206, 216, 217, 270, 271, 331, 334
Component-based, 92, 203, 235–236, 267–268, 288, 297–299, 329, 332–333
Composability, 102–103, 108, 109, 121–122, 132, 216, 265, 267, 278, 293
Composite interfaces, 93–94, 108
Computational cluster, 2, 82, 97
Computer system, 2–10, 13–15, 18–21, 25, 26, 38, 41–45, 47, 49, 52, 53, 56, 59, 60, 62, 63, 65, 80, 85, 95, 96, 112, 127, 136, 141, 144, 161, 163, 164, 176, 220–222, 242, 258, 260, 266, 271–273, 276, 277, 280, 281, 283, 292, 294, 298–301, 305, 320, 322, 328, 329
Concept, 22, 33, 35–41, 44, 48, 58, 81, 84, 88, 99, 101–102, 256, 267, 269, 300, 315, 328
  formation, 36–38
Conceptualization, 36, 38, 41, 43, 45, 97, 108, 113, 136–141, 160, 192–193
Conceptual landscape, 31–33, 35–40, 47, 49, 308
Concurrency control field (CCF), 225, 226
Configuration, 19, 26, 81, 92, 142–143, 156–157, 165, 171, 179, 181, 182, 235, 314, 322, 331, 333, 334, 336
Consistent failure, 87, 140, 176
Contact bounce, 227, 228
Control interface, 92–95
Controlled object, 2–9, 14, 15, 17, 22, 25, 52, 65, 80, 96, 112, 115, 117, 118, 121–123, 164, 194, 219, 227, 229, 230, 233, 265, 266, 274, 284, 288
Controller Area Network (CAN) protocol, 167

Control loop, 6–10, 20, 24–26, 52, 63, 66, 112, 130, 149, 159, 168, 184, 227, 236, 255, 269, 329
Control of pace, 14
Control structure, 23, 138, 218, 222, 223, 228, 235, 236, 247, 294, 296
Convergence function, 67, 69–72, 76
Crash failure, 11, 115, 139–140, 305
Critical instant, 103, 178, 252
Criticality level, 143, 144, 272, 276, 280–281
Cryptographic methods, 142, 143, 145–148, 287, 318
C-task. *See* Complex task
Cyber-physical, 2, 26, 52, 59, 308
Cyberspace, 128, 142, 305, 306, 308, 309, 318, 319, 321, 322, 328

D
Database components, 88, 284–285
Data collection, 3–5, 88
Data efficiency, 174, 222
Datagram, 173, 180
Deadline, 3, 14, 15, 21, 25, 93, 130, 220, 228, 236, 240–243, 246, 248–253, 255–257
Dead time, 8, 9, 24–25, 66, 130, 168, 184, 227, 236, 255, 329
Delay jitter, 8, 9, 25–26, 69, 113–114, 123, 182, 184, 187, 329, 331
Delivery order, 53, 57–58, 76, 182
Denial of service, 106, 142, 145, 318, 320, 322
Dense time, 41, 59, 62–66, 97, 125–126
Dependability, 10–13, 19, 42, 135–166, 168–171, 260, 270, 276–279, 282, 302–303, 327–328, 331–332, 337
Derived property, 44, 57
Design diversity, 281–284
Design for testability, 293–294, 305
Determinism, 31, 119, 125–130, 132, 158, 170–171, 181, 184, 186, 220, 226, 236, 294, 326
Development cost, 17, 21, 26, 263, 292, 305
Device scaling, 200–203, 211, 212
Difficult task, 34, 45, 46, 49, 172, 250, 312
Digital input/output, 227–228
Digitalization error, 54, 59, 61, 233
Digital signature, 147, 319
Direct digital control, 3, 5–6
Diverse software, 281–284
DO–254, 280
Domain expert, 32
Drift offset, 67, 72, 73, 76
Drift rate, 54–56, 66, 67, 70, 72–74, 76
Dual role of time, 221–223
Duration, 2, 4, 7, 8, 23, 24, 42, 52–54, 56–62, 65, 66, 70, 76, 83, 87, 89, 90, 123–125, 130, 149, 155, 159, 162, 164, 168, 174, 175, 181–182, 184, 223, 228, 255, 268, 293, 305
Dynamic scheduling, 17, 177, 219–220, 240, 242, 243, 251–255, 333, 335

E
Earliest-Deadline-First (EDF) algorithm, 252–253, 257
Electronic product code (EPC), 312, 313, 316–317, 319, 321
Elementary interfaces, 93, 94, 108
Elevator example, 17
Embedded real-time systems, 17–20, 49, 262–263, 265, 268, 269
Emergence, 43–45, 49
Emergent behavior, 44, 105, 106, 298
Encapsulation, 38–39, 81, 204, 205, 207, 231, 265, 297, 334
Encryption, 142, 145–149, 164, 318
Endianness, 96, 208
End-to-end error detection, 304
End-to-end protocol, 22, 170
Energy estimation, 191, 193–197, 208
Energy harvesting, 191, 209–212, 317, 320, 321
Engine control, 19, 23–24, 27, 28, 81, 93, 119
Entity, 32, 96, 105, 106, 308, 331
EPC. *See* Electronic product code
Error containment region, 157, 167, 205
Error detection, 14, 39, 91, 103, 135, 141, 152–153, 169, 170, 180, 184–185, 188, 231, 234–235, 304, 305
  coverage, 15, 27, 139, 229, 273, 283, 304, 326
  latency, 10, 139, 176, 188, 271, 329
ESTEREL, 83
ET. *See* Event-triggered
Event, 2, 30, 52, 84, 113, 138, 173, 194, 218, 248, 266, 295, 315, 326
Event-triggered (ET), 4, 13, 16–17, 25, 79, 84, 90, 91, 100, 110, 115, 124, 138, 167, 169, 173, 178–180, 183–188, 218–220, 223, 224, 255, 294, 298, 331, 334
Exact voting, 157
Execution time, 83, 84, 107, 114, 118, 123, 129, 152, 204, 206–208, 212, 218–221, 225, 234, 240, 241, 244–251, 257, 332
Explicit flow control, 175–177, 188
Explicit synchronization, 130, 218, 226, 236, 247
External clock synchronization, 73–76, 335

F
Fail-operational, 13, 15, 161, 272, 273, 278, 283
Fail-safe, 13, 15, 161, 272, 278, 282–283
Fail-silent, 69, 139, 155–156, 159, 161, 185, 230, 235, 331
Fail-stop, 139–140
Failure, 1, 47, 55, 80, 111, 135, 169, 199, 218, 241, 259, 293, 311, 331
  detection, 140, 151–153, 155, 165
  rate, 10–12, 26, 27, 137, 154, 155, 164, 165, 199, 201, 232, 271, 272, 276, 278, 284–286
Failure in time (FIT), 10, 155, 164, 201
Fate-sharing model, 330–331
Fault containment unit (FCU), 46, 47, 136–137, 140, 151, 154–158, 160, 165, 271, 278, 286, 315, 329, 331
Fault hypothesis, 11, 16, 135, 153–155, 159, 165, 173, 178, 184, 185, 295, 303
Fault injection, 140, 152–153, 235, 277, 291, 295, 302–306, 327
Fault isolation, 172
Fault tolerance, 52, 74, 81, 124, 127, 153–159, 165, 168, 171, 178, 221, 262, 265, 270, 277–279, 286–287, 295, 302, 303
Fault-tolerant actuators, 229–231
Fault-tolerant average algorithm, 68, 77, 185
Fault-tolerant unit (FTU), 73, 74, 155–159, 163, 165, 172, 271, 281, 331
Fault tree analysis, 274, 275
FCU. *See* Fault containment unit
Feedback scheduling, 256–257
Firewall, 144, 169, 312, 335
Firm deadline, 3, 25
FIT. *See* Failure in time
FlexRay, 187
Flow control, 22, 39, 83, 91, 93, 107, 174–177, 179, 180, 188, 293, 298
Formalization, 299–300
Formal methods, 280–282, 293, 299–302
Frequency scaling, 202, 207, 216, 268, 335
FTU. *See* Fault-tolerant unit

G
Gateway component, 40, 46, 48, 82, 95–99, 104, 108, 269–270, 318, 330, 335, 337
Gateway services, 335
GENeric Embedded SYStems (GENESYS), 104, 325, 327–328, 336
Generic middleware (GM), 216, 217, 332–335, 338
GENESYS. *See* GENeric Embedded SYStems
Global positioning system (GPS), 57, 73, 74, 107, 308, 332, 335
Global time, 17, 47, 51–78, 84, 107, 110, 113, 114, 116, 124, 128, 152, 155–156, 167, 168, 173, 184, 186, 218, 235, 329, 331, 332, 334
Global time base, 47, 52, 53, 59, 61, 65, 66, 73, 74, 114, 121, 124, 126, 127, 136, 157, 158, 167, 168, 176, 221, 222, 326–328, 337
GM. *See* Generic middleware
GPS. *See* Global positioning system
Granularity, 17, 54, 55, 58–61, 63, 64, 86, 116, 124, 126, 131, 206–207, 222, 294
Ground state, 86–88, 107, 155, 162–163, 294, 331
g-state recovery, 87–88, 163
Guaranteed response, 16, 26

H
Hamming distance, 152, 179
Hard deadline, 3, 21, 25, 240, 250, 251
Hard real-time, 3, 10, 11, 13–16, 21, 25, 26, 176, 240, 243, 247, 250, 251, 255, 271–272, 295
Hazard, 142, 245, 274–277, 288
Heisenbug, 126, 138, 329, 336
Hidden channel, 122, 123
High-level protocols, 172, 173, 187, 203, 217, 219, 314, 333

I
Idempotency, 122–125
IMA. *See* Integrated modular avionics
Implicit flow-control, 176, 188
Inconsistent failure, 140
Indirect observation, 114
Industrial plant automation, 20, 21
Inexact voting, 157–158
Information pull, 93, 108, 109, 270
Information push, 93, 109, 270
Information security, 141–149, 164, 312, 318, 322
Input/output, 22, 92, 94, 200, 215, 216, 226–232, 240, 270
Instant, 2, 34, 52, 80, 112, 136, 168, 218, 241, 286, 303, 329
Instrumentation interface, 2–3
Integrated modular avionics (IMA), 281
Integration viewpoints, 104

Intermediate forms, 46, 265
Intermittent fault, 137
Internal clock synchronization, 66–73, 116
International Atomic Time (TAI), 56–57, 73, 75, 76
Internet of Things (IoT), 13, 20, 107, 142, 145, 307–322
Interoperability, 98, 99, 101, 103, 309
Interprocess communication, 216–217
Interrupts, 17, 70, 84–86, 90, 93, 115, 129, 169–170, 178, 218–220, 228–229, 235–237, 240, 243, 245–249, 255, 334
Interval measurement, 59–60, 73
Intrusion, 141, 144, 149, 150, 164, 308
Intuitive-experiential, 30, 47, 49
Invisible information flows, 34
IoT. *See* Internet of Things
Irreducibility, 43
Irrevocable action, 15, 123

J
Jitter, 8–10, 23–26, 69–71, 77, 113–114, 116, 121, 123, 125, 152, 168, 169, 172, 173, 178, 180, 182, 184, 187, 188, 222, 255, 326, 329, 331

K
KERBEROS, 146, 149
Key management, 146

L
Latency jitter, 9, 23, 25, 69, 70, 77, 178, 188
Layering, 172, 182, 217, 314
L-determinism, 126, 129
Least-laxity (LL), 252–253, 257
Legacy system, 105
LIF. *See* Linking interface
Life-cycle cost, 17, 19
Linking interface (LIF), 46, 92, 94–104, 106, 108, 109, 156, 216, 217, 265, 267, 270, 296–298, 329–332, 334, 335, 337
LL. *See* Least-laxity
Load hypothesis, 16, 26
Local interface, 92–98, 101, 104, 108, 217, 265, 270
Logical control, 82–84, 107, 207
Low-level protocols, 173, 178–179, 187
Low-power hardware design, 201

M
Maintainability, 12, 26, 265, 284–287, 332
Maintainable Real-time System (MARS) project, 326–327
Maintenance strategy, 19, 284–287
Malicious failure, 140, 158, 270
Malign failure, 11, 26, 140
Man–machine interface, 2, 24, 81, 98, 152, 171, 187, 265
Mark method, 53
MARS project. *See* Maintainable Real-time System project
Mean time between failures (MTBF), 12–13
Mean time to a failure (MTTF), 10–13, 201, 272, 293, 305
Mean time to repair (MTTR), 12–13, 138
Measured data, 5, 233, 234, 236
Media-access protocols, 69–70, 129
MEDL,
Membership, 116, 154, 158–159, 165, 170, 184–185, 187, 188, 334
MEMS. *See* Micro-Electro-Mechanical Systems
Message, 10, 35, 62, 80, 113, 139, 168, 194, 216, 250, 262, 293, 311, 329
Meta-level specification, 99, 101–103, 108
Microarchitecture, 204, 245–246
Micro-Electro-Mechanical Systems (MEMS), 231, 320
Microtick, 54–56, 58–62, 66, 73, 75, 76
Model, 2, 30, 52, 80, 114, 136, 171, 193, 218, 247, 263, 292, 315, 326
Monolithic system, 105–106
MPSoCs. *See* Multiprocessor systems on chips
MTBF. *See* Mean time between failures
MTTF. *See* Mean time to a failure
MTTR. *See* Mean time to repair
Multicasting, 80–81, 180, 293
Multimedia, 21–22, 87, 99, 140, 160, 162, 180, 182–183, 208, 225, 256, 257, 328, 329
Multiprocessor systems on chips (MPSoCs), 104, 155, 197, 198, 204–206, 209, 248, 263, 336–337

N
Naming, 40, 95–97, 108, 312–313, 318, 321, 334
NBW protocol. *See* Non-blocking write protocol
NDDC. *See* Non-deterministic design constructs
Near field communication (NFC), 312–314

Network authentication, 146, 148–149
Network time protocol (NTP), 75, 77
NFC. *See* Near field communication
Non-blocking write (NBW) protocol, 224–226
Non-deterministic design constructs (NDDC), 127–130, 132, 331, 333, 338
Non-preemptive, 219, 240, 243, 246, 255
NTP. *See* Network time protocol

O
Object delay, 8, 9
Observation, 2, 32, 52, 81, 113, 144, 168, 221, 253, 297, 311
Occam's razor, 36, 48
Omniscient observer, 54, 60, 123
One-way functions, 319, 322
Open-loop control, 20
Operating system, 44, 69, 70, 86, 91, 94, 142, 152, 162, 164, 177, 202, 204, 207–208, 215–236, 243, 245–247, 249, 263, 268, 332–333, 338
Operational specification, 99–103, 108

P
Parametric RT image, 119, 120
Partitioning, 2, 12, 19, 35, 43, 46, 48, 49, 65, 83, 92, 152, 163, 187, 204, 248, 268–269, 272, 278, 281, 283, 293, 333
Password attack, 145
Pathfinder, 153, 255
Peak-load, 1, 5, 14, 16, 25, 26, 223, 235, 295
Perfect clock, 55
Periodic tasks, 241–243, 250, 253, 256, 257
Permanence, 111, 122–125
Permanent failure, 156, 164, 165, 169, 199, 201, 211, 286
Phase-aligned transaction, 65–66, 118, 120, 329
Physical one-way functions (POWF), 319, 322
PIM. *See* Platform-independent model
Plant automation system, 17, 20–21, 24, 26
Platform-independent model (PIM), 92, 203, 204, 268, 269, 296, 299
Platform-specific model (PSM), 92, 203, 204, 296, 299
Pocket calculator, 85–86
Pollack's rule, 204–205, 212, 262–263
Positive acknowledgment-or-retransmission (PAR) protocol, 169, 175, 177
Power blackout, 5, 142, 326
POWF. *See* Physical one-way functions
Precedence graph, 248–250
Precision, 23, 37, 39, 51, 55–56, 58, 60, 65–72, 76, 77, 90, 99, 110, 116, 121, 126, 152, 168, 173, 176, 183, 184, 186, 187, 206–207, 221, 222, 234, 248–249, 332, 334
Preemptive scheduling, 129, 240–241, 256
Primary event, 5, 53
Principles of composability, 79, 102–103, 108
Priority ceiling protocol, 239, 246, 253–255, 257
Privacy, 13, 141, 147, 164, 307, 308, 310, 314, 318–320, 322
Private key, 146–148
Probe effect, 47, 151, 270, 293, 294, 298–299, 305, 315, 331
Problem solving, 29–34, 45, 46, 48, 260
Process lag, 8, 25
Propagation delay, 174, 175, 180–182
Property mismatches, 96–98, 106, 108, 208, 318, 330, 335
Protocol, 22, 39, 51, 96, 114, 146, 167, 197, 215, 239, 312, 326
  latency, 167
PSM. *See* Platform-specific model
Public key, 146–149, 164
Purpose and viewpoint, 41–42

R
Radio Frequency Identification (RFID), 307, 309–312, 315–322
  security, 318–320
  tags, 210, 309–310, 312, 316–320
Rare event, 5, 14, 16, 153, 165, 177, 255, 257, 266–267, 295, 302, 326
Rate constrained, 89, 167, 173, 176, 178, 180–183
Rate correction, 72–75, 77
Rate monotonic algorithm, 239, 251–252, 254, 257
Raw data, 4, 233
Real-time (RT), 1, 39, 51, 79, 111, 139, 167, 202, 215, 239, 259, 292, 310, 325
  entity, 3–5, 8, 9, 25, 26, 111–122, 125, 130–131, 168, 169, 171, 187, 194, 221, 226–227, 229–230, 233
  image, 4, 5, 26, 111, 112, 115–120, 124, 125, 131, 194, 231
  object, 111, 112, 116, 120, 122, 125, 131, 194
  systems, 1, 43, 51, 79, 124, 152, 167, 206, 215, 239, 259, 293, 325
Reasonableness condition, 58–60, 76
Recovery of determinism, 130
Redundant sensors, 157, 233–234

Reference clock, 51, 54–61, 63, 116–117, 206
Reintegration cycle, 87, 162, 164
Reintegration point, 86–87, 161–162, 164, 216, 294
Reliability, 10–12, 18, 20, 26, 43, 44, 115, 137, 153–155, 165, 167–169, 172, 173, 178, 180, 188, 191, 197–199, 201, 203, 211, 269, 271, 272, 275, 276, 278, 281, 282, 285, 286, 297, 314, 336
Replica determinism, 119, 127, 129, 130, 158, 171, 220, 226, 236
Resource adequacy, 11, 13, 16
Response time requirements, 14, 20, 228
Resynchronization interval, 67, 72, 76
Rise time, 7–9, 25, 41, 227
Risk, 20, 57, 274, 277, 279, 288, 310, 326
Rolling mill, 24–25, 82, 109–110
RT. *See* Real-time
RTCA/DO–178B, 280, 281

S
Safety, 11–12, 14, 15, 19, 20, 26, 142, 151, 161, 165, 247, 271–283, 292, 310–312, 333, 337
  analysis, 143, 274–276, 280, 288
  case, 276–280, 288, 305
  standards, 279–281
Safety-critical systems, 3, 6, 11, 14, 94, 151–153, 165, 172, 233, 271–272, 274, 277, 279, 281–284, 293, 333
Safety integrity level (SIL), 266, 279, 280
Sampling period, 8–10, 220
Schedulability test, 241–242, 248, 253–256
Schedule period, 184, 241, 248, 249
Scheduling dependent tasks, 246, 253–255
Scientific concepts, 37–38
Search tree, 249
Security, 13, 19, 26, 106, 141–149, 164, 205, 310, 312–314, 318–320, 322, 329, 332, 334–337
  threats, 143–145, 318, 322
Segmentation, 19, 35, 47, 48, 70, 83, 84, 208
Self checking component, 140
Semantic agreement, 233–234
Semantic content, 4, 37, 39–40, 48, 92, 95, 97, 98, 112
Server task, 250–251, 314, 322
Service, 10, 39, 66, 80, 114, 136, 168, 195, 216, 240, 260, 293, 308, 328
Signal conditioning, 4, 5, 26, 231–233
SIL. *See* Safety integrity level
Simple task (S-task), 45, 46, 218–220, 243–247
Simplicity, 29–49, 328
Simplification strategies, 34–35
Simultaneous events, 52, 127, 223
SOC. *See* Sphere of control
Soft real-time, 3, 13–15
Software component, 80, 216, 218, 236, 268, 270, 297, 329, 332, 333, 338
Software maintenance, 287
SoS. *See* System of system
Source code analysis, 244
Space-time lattice, 64–65
Sparse time, 62–66, 83, 85, 101, 128, 184, 240
Sphere of control (SOC), 4, 8, 9, 62, 65, 105, 106, 112, 175, 227, 228
Spoofing attack, 145
SRU, 27, 28
Standardized message interfaces, 98
S-task. *See* Simple task
State, 2, 34, 52, 80, 112, 136, 169, 192, 216, 242, 261, 294, 326
  estimation, 116, 118, 120–122, 124, 131, 164, 329
Stateless voter, 230–231
Stochastic drift rate, 72
Sub-threshold logic, 202, 206
Sufficient schedulability test, 241, 248, 253, 254, 256, 257
SUT. *See* System-under-test
Synchronization condition, 66–69, 72, 76
Synchronized actions, 118
Syntactic agreement, 233, 234
System C, 92, 203, 269
System design, 14, 16, 92, 103, 146, 161, 209, 248, 251, 259–289, 326
System evolution, 45, 104–106, 296–297, 322
System of system (SoS), 40, 46, 104–108, 130, 332
System-under-test (SUT), 292–297, 305

T
TADL. *See* Task-descriptor list
TAI. *See* International Atomic Time
Task, 5, 30, 75, 82, 118, 138, 172, 202, 217, 240, 280, 295, 312, 326
Task-descriptor list (TADL), 218–219, 236
TDI. *See* Technology dependent debug interface
TDMA. *See* Time division multiple access
Technology-agnostic, 94, 203–204, 212
Technology dependent debug interface (TDI), 92–93, 95, 216, 217

Technology independent control interface (TII), 92–94, 163, 216, 217, 297–298, 333, 334, 337
Testability, 126, 270, 293–294, 305
Test coverage, 294, 295
Test data selection, 294–295
Testing challenges, 293–297, 305, 306
Test of a decomposition, 269–271
Test oracle, 296, 305
Thermal effects, 195, 197–199, 212
Thrashing, 176–177
Three Mile Island, 170
Throughput-load dependency, 176–177
TII. *See* Technology independent control interface
Time division multiple access (TDMA), 110
Time-stamp, 17, 47, 52, 54, 57–59, 61–64, 68–70, 90, 113–117, 122, 127, 222, 228, 332
Time-triggered (TT), 4, 9, 13, 16–17, 25, 66, 70, 75, 77, 84, 91, 100, 101, 104, 107, 108, 114, 115, 121, 124, 138, 156, 159, 163, 173, 178, 179, 183–188, 218–220, 222–224, 229, 236, 248, 255, 269, 281, 293, 294, 296, 325–338
Time-triggered ethernet (TTEthernet), 179, 186, 188, 223, 331, 338
Time-triggered network on chip (TTNoC), 330, 331, 336–337
Time-triggered protocol (TTP), 101, 154, 184–185, 187, 188, 327, 331, 338
TMR. *See* Triple-modular redundancy
TNA. *See* Trusted network authority
Token bus, 181
Top event, 274, 275
Transient fault, 137, 152–154, 161, 205, 230, 235, 271, 276, 331, 336
Transitory fault, 137
Transport specification, 99–100, 103, 108, 298–299
Trigger, 4, 16–17, 25, 80, 83, 84, 90, 91, 123, 218, 221, 268, 274, 288
  task, 220, 222, 228
Triple-modular redundancy (TMR), 155–157, 165, 230, 231, 236, 278, 283, 284, 337
Trusted network authority (TNA), 336–337
Trusted security server, 146–149
TT. *See* Time-triggered
TTEthernet. *See* Time-triggered ethernet
TTNoC. *See* Time-triggered network on chip
TTP. *See* Time-triggered protocol
Two-faced failures, 140

U
UDP. *See* User datagram protocol
Ultra-high dependability, 277, 282
UML. *See* Unified modeling language
UML MARTE. *See* Unified modeling language MARTE
Understanding, 20, 31–34, 36–38, 40–41, 43, 45–48, 81, 95, 102, 104, 111, 126, 130, 193, 234, 260, 261, 264–268, 281–282, 288, 309, 326, 328, 329, 336
Unified modeling language (UML), 265, 268
Unified modeling language (UML) MARTE, 203, 265, 268–269
Universal time coordinated (UTC), 56, 57, 75–77
User datagram protocol (UDP), 180
UTC. *See* Universal time coordinated

V
Validation, 5, 11–12, 263, 264, 277, 280, 281, 291–306
Value failure, 125, 139, 156
VOTRICS, 282, 283
Vulnerability, 141, 142, 144, 150, 164, 287–289, 318, 319

W
Waistline model, 172–173, 187
Watchdogs, 15, 159, 235, 255
WCAO. *See* Worst-Case Administrative Overhead
WCCOM. *See* Worst-case communication
WCET. *See* Worst-case execution time
Wireless sensor networks (WSN), 320–321
Worst-Case Administrative Overhead (WCAO), 219, 243–246
Worst-case communication (WCCOM), 118–120
Worst-case execution time (WCET), 83, 118–120, 152, 202, 208, 219–221, 225, 228, 234, 236, 243–248, 255, 256, 269, 283, 295
WSN. *See* Wireless sensor networks