FP7-ICT-2013.3.4

Project acronym:	**AMADEOS**
Project full title:	Architecture for Multi-criticality Agile Dependable Evolutionary Open System-of-Systems
Grant Agreement no.:	610535

Partners:

[1] Università degli Studi di Firenze

[2] Technische Universitaet Wien

[3] Universite Joseph Fourier Grenoble 1

[4] ResilTech S.r.l.

[5] Thales Nederland Bv

[6] European Network For Cyber Security Cooperatief Ua

D2.1 - BASIC SOS CONCEPTS, GLOSSARY AND PRELIMINARY CONCEPTUAL MODEL

(01.12.2013-30.06.2014)

Revision of the document:	05
Responsible partner:	TUW
Contributing partner(s):	all
Reviewing partner(s):	UNIFI, UJF
Document date:	24/06/2014
Dissemination level:	PU

Change Records

Revision	Date	Changes	Authors	Status[1]
00	18/03/2014	Document layout, Introduction, Fundamental System Concepts, Time, Data and State, Actions and Behaviour, Communication, Emergence, Problem Solving.	TUW	NV
01	26/03/2014	Consolidated all contributions to: Dependability and Security, Evolution and Dynamicity, System Design and Tools, Governance, and Quality Metrics and Attributes.	all	NV
02	05/05/2014	Provided motivation and context for introduced concepts. Comments/Feedback for plenary meeting.	all	NV
	01/06/2014	Incorporated feedback from plenary meeting.	all	NV
03	01/06/2014	First round of review version.	TUW	NV
	11/06/2014	Addressed review comments w.r.t. Section 9.	ENCS	NV
	16/06/2014	Addressed review comments w.r.t. Section 11.	TNL	NV
	16/06/2014	Addressed review comments w.r.t. Sections 2, 4. Improved overall document consistency. Added glossary.	TUW	NV
04	17/06/2014	Second round of review version.	TUW	NV
	24/06/2014	Updated executive summary. Addressed final review comments.	TUW + ENCS, TNL, RES	NV
05	24/06/2014	Final version.	TUW	V

[1] V - Valid, NV - Not Valid, R - Revision

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Definitions and Acronyms

Acronym	Definition
CP	Consume/Produce
CPS	Cyber-Physical System
CS	Constituent System
DoW	Description of Work
ET	Event-Triggered
GLONASS	Globalnaja nawigazionnaja sputnikowaja Sistema – Global Satellite Navigation System
GPS	Global Positioning System
GPSDO	GPS Disciplined Oscillator
IoD	Interval of Discourse
IoT	Internet of Things
MCDA	Multi-Criteria Decision Analysis
OODA	Observe, Orient, Decide and Act
PAR	Positive Acknowledge or Retransmission
RT	Real-Time
RUMI	Relied Upon Message Interface
RW	Read/Write
SBR	Scenario-Based Reasoning
SoC	Sphere of Control
SoS	System-of-System
TAI	Temps Atomique International
TT	Time-Triggered
UoD	Universe of Discourse
UTC	Universal Time Coordinated
WCET	Worst Case Execution Time

Executive Summary

This preliminary version of the AMADEOS conceptual model is the result of the joint work in WP 2 during the first six months of AMADEOS. As outlined in the Description of Work (DoW) this preliminary conceptual model will be further discussed and refined in the remaining two years of the AMADEOS project to arrive at a final conceptual model at the end of the project.

The report starts with an elaboration on the nature of concepts and conceptual modelling in general. A concept captures a *unit of thought* that lifts the mental processes to a *higher level of effectiveness.* In the history of the sciences, the formation of appropriate concepts has been a necessary prerequisite for the formulation of novel natural laws and thus the advancement of science. The aim of a conceptual model is to explain the meaning of the concepts and the associated terms used by domain experts in a field of science and to describe the correct relationships between the different concepts.

After the short introduction, this report introduces twelve categories of concepts that deal with the following System-of-System (SoS) topics: (2) *Fundamental System Concepts*, (3) *Time*, (4) *Data and State*, (5) *Actions and Behavior*, (6) *Communication*, (7) *Emergence*, (8) *Problem Solving*, (9) *Dependability and Security*, (10) *Evolution and Dynamicity*, (11) *System Design and Tools*, (12) *Governance*, (13) *Quality Metrics and Attributes*. In the final Section of the Report a Glossary lists the concise definition of the concepts in alphabetical order.

Conceptual modelling is a continual process that has to go through many iterations in order to arrive at a consolidated level. This report documents the first baseline of System-of-Systems (SoS) concepts that will be used and tested within the next phases of the AMADEOS project, and possibly even within a wider part of the SoS community, to express relevant SoS properties. It is expected that during this usage the appropriateness of some concepts will be confirmed, other concept definitions will have to be revisited and new additional concepts will have to be introduced. At the end of the AMADEOS project we expect to deliver a mature conceptual model that has already passed a rigorous acceptance test.

1 INTRODUCTION

A recent (2007) report on *Software for Dependable Systems: Sufficient Evidence?* from the US National Academies [12] contains as one of its central recommendations*:*

One key to achieving dependability at reasonable cost is a serious and sustained commitment to simplicity, including simplicity of critical functions and simplicity in system interactions. ***This commitment is often the mark of true expertise.***

We feel that the first important step of achieving simplicity is the development of an understandable conceptual model of a domain. If the language used to talk about a domain contains terms with clearly defined meanings that are shared by a wide part of the community working in the domain, then the communication of ideas among experts is simplified. We hope that this AMADEOS conceptual model of a System-of-Systems (SoSs) contributes towards such a clarification.

An SoS comes about by the message-based integration of existing systems (legacy systems), normally operated by different organizations and new systems that have been designed to take advantage of this integration. The following table compares some of the characteristics of an SoS compared to those of a monolithic system.

Table 1: Comparison of an SoS compared to a monolithic system

Characteristic	Monolithic System	SoS
Scope of System	Fixed (known)	Not known
Clock synchronization	Internal	External (GPS)
Structure	Hierarchical	Networked
Requirements and Spec.	Fixed	Changing
Evolution	Version control	Uncoordinated
Testing	Test phases	Continuous
Implementation Technology	Given and fixed	Unknown
Faults (Physical, Design)	Exceptional	Normal
Control	Central	Autonomous
Emergence	Insignificant	Important
System development	Process model	???

If we look at this table we realize that many of the established assumptions in classical system design, such as e.g., the *scope of the system is known*, that *the design phase of a system is terminated by an acceptance test* or *that faults are exceptional events,* are not justified in an SoS environment.

SoS engineering is closely related to many other IT domains by looking at the *glue* that is needed to integrate the diverse systems:

- *Cyber-Physical Systems (CPSs)* and *embedded systems* that deal with the integration of systems that consist of a physical subsystem and a cyber subsystem.
- *Internet of Things (IoT)* that investigates the integration of an enormous variety of small smart sensors and large CPSs via the Internet.
- *Big Data* that looks at the analysis of the enormous, often heterogeneous data streams that are captured by the *IoT* and *SoSs*.

The concepts introduced in AMADEOS to describe the properties of an SoS can thus form a foundation of a conceptual model in these other domains as well.

In the DoW of AMADEOS it is stated in the first line of the abstract *The objective of this research proposal is to bring time awareness and evolution into the design of System-of-Systems (SoS).* The notion of *time* takes thus a prominent position in our understanding of an SoS and has influenced many of the presented concepts.

We start our work on the *conceptual model of an SoS* by looking at the *nature of concepts* and try to establish an agreement about the meaning of the term *conceptual model*.

1.1 ON THE NATURE OF CONCEPTS

In a changing world, *knowledge* about essential and permanent properties of objects and situations must be acquired and maintained since such knowledge is of critical importance for human survival. The continuous and immense flow of direct and indirect sensory data (eyes, ears, touch, smell, ...) to the human mind requires effective mechanisms to filter out those impressions that are considered relevant and must be stored in the brain to construct a useful *image* of the environment [13]. The most important of these mechanisms is the *mechanism of abstraction*. Abstraction is a process where the *characteristic particulars of an impression* are recognized and remembered to establish and identify a new category that is of later use in the construction of a mental model of reality.

Take the example of face recognition: Even a small child can remember, after a brief look at a person, the characteristic features of a face such that a person can be recognized again at a later time, even if the environmental conditions (lightning, line of sight) have changed. This very effective process of abstraction is carried out in the *intuitive experiential subsystem* of the human mind without any guidance by the *rational subsystem* [2, p.30].

Abstraction forms categories that are, considered in isolation, neutral. In order to establish a *utility* of a category, it must be linked with other categories. In the previous example, the *recognized person* can be linked with the category *friend* or *enemy*.

> ***Concept: A category that is augmented by a set of beliefs about its relations to other categories, i.e., existing knowledge, is called a concept.***

- The set of beliefs relate a *new concept* to already *existing concepts* and provides an *implicit theory* (a mental model) of the domain.
- The theory explains how the individual interconnects the diverse concepts of the domain and *understands* their interrelationships.
- As a new domain is penetrated, new concepts are formed and linked to already existing concepts.

A new concept establishes a new *unit of thought* that lifts the mental processes to a *higher level of effectiveness. A new concept emerges and takes shape in the course of a complex operation aimed at the solution of some problem [17, p.54].* Problem solution and concept formation are thus closely related. In most cases, the formation of a new concept leads to an *abrupt simplification of a problem scenario*. In the history of the sciences, the formation and introduction of appropriate new concepts, such as *mass*, *acceleration,* etc., have been a necessary prerequisite for the formulation of novel natural laws and thus the advancement of science.

Normally a *name* (sound, string) or a *language phrase*—in scientific jargon— an *abbreviation,* is assigned to a new concept. The name can only be understood (i.e. has a meaningful content to a user) *after* the concept has been formed.

A concept requires for its formation a number of experiences that have something in common:

- abstracting from familiar examples (*most important*) — starting with primary concepts (*ostensive* definition).
- prototype—an entity that depicts the typicality of the concept.

- feature specification (features may not always be *necessary* nor *sufficient*).
- establishing a set of *beliefs* and *relations* with already existing concepts (*concepts as theories*).
- precise definition in a language (*essential* definition).

Frequently a *precise definition* of a concept is not possible, since many concepts become fuzzy at their boundaries.

There is a *natural level of categorization*, neither too specific nor too general, that is widely used in thinking and conversation. This categorization leads to *basic-level concepts*. Basic level concepts are usually represented in the language by a *single* word. Take the example of the basic level concept *chair*, which is more concrete than *furniture*, but more abstract than *arm-chair*. Studies with children have shown that basic-level concepts are *acquired earlier* than *sub-concepts* or *encompassing* concepts.

Another classification of concepts distinguishes between *primary concepts* that are those directly derived from sensory experience (e.g., warm, cold) *and secondary concepts* that are those abstracted from other concepts, e.g., the example *furniture* from above. *Basic-Level* concepts are not necessarily *primary* concepts. By forming more and more abstract concepts on top of lower level concepts, a hierarchy of concepts can be constructed, leading to what [18, p.103] calls an *abstraction ladder*.

A new domain can be explained most effectively if the issues are treated at differing levels of the abstraction ladder. The starting point is a set of examples and observations based on already acquired concepts. The introduction of new, *more abstract concepts* must be based on generalization and abstraction from those already familiar existing concepts. Abstract analysis and concrete interpretation and explanation must be intertwined frequently. If one remains only at a *given low-level of abstraction,* then the amount of non-essential detail is overwhelming. If one remains only at a *given high-level of abstraction*, then relationships to the world as it is experienced by the senses are difficult to form.

The *knowledge base* of personal concepts that has been built up and is maintained by an individual in the *experiential* and *rational* subsystem of the mind over the lifetime of a human is called the *conceptual landscape (Weltbild)* [2, p.35]. This conceptual landscape is the result of genetic traits, sensory experience and communication. *Understanding a new concept* means that the properties of the *new concept* can be linked with already existing concepts in the *conceptual landscape* of the observing human. The firmer these links are, the better the understanding.

1.2 CONCEPTUAL MODELING

According to [17], a conceptual model is characterized as follows:

The aim of a conceptual model is to express ***the meaning of terms and concepts*** *used by domain experts to discuss the problem, and to find the correct relationships between different concepts. The conceptual model attempts to clarify the meaning of various, usually ambiguous terms, and ensure that problems with different interpretations of the terms and concepts cannot occur. Such differing interpretations could easily cause confusion amongst stakeholders, especially those responsible for designing and implementing a solution, where the conceptual model provides a key artifact of business understanding and clarity. Once the domain concepts have been modeled, the model becomes a stable basis for subsequent development of applications in the domain.*

We discussed this characterization of a conceptual model at length within the AMADEOS project and came to the conclusion to support it fully. This characterization of a conceptual model, which is in agreement with wide opinions within our community, is thus forming a reference for the AMADEOS work on developing a conceptual model for an SoS.

A conceptual model establishes a domain specific ontology, since ontology *is a specification of the conceptualization of a domain of discourse.* It outlines a *domain specific vocabulary* and defines an *implicit theory* about a domain of discourse. An *ontological* commitment is the agreement to use the *shared vocabulary* in a coherent and consistent manner.

1.3 STRUCTURE OF THIS DOCUMENT

In this document concepts that are considered relevant for the description of an SoS are structured along a set of high level topics. The definition of a concept is presented in ***italics bold*** while the discussion of the definitions is expressed in standard text.

As mentioned before, some circularity in the definitions is unavoidable, since some agreement on the meaning of words and phrases in the natural language must be assumed *a priori*. We have tried to keep this circularity to a minimum. We have also tried not to introduce formalisms whenever possible. In our opinion, a formalism is only of value after a clear understanding of a concept has been gained by understanding the concept in its natural language presentation. Replacing fuzzy concepts by Greek letters does not improve the understanding.

At the end of the document we list all definitions in alphabetical order and point to the Section of the document, where the definition has been introduced.

2 FUNDAMENTAL SYSTEM CONCEPTS

Our philosophical view—termed *scientific realism* [18] — is committed to a *mind-independent world* that can be investigated by the sciences.

2.1 BASIC CONCEPTS

We start by delineating the universe and time of discourse of an *SoS*.

Universe of Discourse (UoD): The Universe of Discourse comprises the set of entities and the relations among the entities that are of interest when modeling the selected view of the world.

The word *domain* is often used as synonym to the notion *Universe of Discourse.*

Interval of Discourse (IoD): The Interval of Discourse specifies the time interval that is of interest when dealing with the selected view of the world.

In order to structure the *UoD* during the *IoD,* we must identify objects that have a distinct and self-contained existence.

Entity: Something that exists as a distinct and self-contained unit.

We distinguish between two very different kinds of entities, *things* and *constructs*.

Thing: A physical entity that has an identifiable existence in the physical world.

Referring to the *Three World Model of Popper* [19] there is another class of entities, we call them *constructs* that have no physical existence on their own but are products of the human mind.

Construct: A non-physical entity, a product of the human mind.

Examples of constructs are *an idea*, *a goal* or any other non-physical entity that is used in communication among humans to express a thought. In the *Three World Mode*l of Popper constructs are, as products of the human mind, in world three.

Humans can create entities—physical or non-physical—which are sometimes called artifacts.

Artifact: An entity that has been intentionally produced by a human for a certain purpose.

An entity can have one or more characteristic qualities—we call such a characteristic quality a *property*—that distinguishes an entity from some other entities.

Property: An instantiation of a characteristic quality.

Properties are concepts that have been introduced in the history of science in order to be able to systematically characterize and investigate things. For example, a *thing*, e.g., a *stone*, can have the property of *mass*, *temperature*, *texture*, etc.. It is often useful to refine a property further and to introduce a quantitative measure for a property.

Attribute: A refinement of a property.

In many areas of science a *metric* has been established that makes it possible to assign a value to the attribute of a property.

Value: An element of the admissible value set of an attribute.

In some circumstances an attribute can be characterized by a numerical value if there is an agreement on the measurement units. For example, we can express the *mass of a stone* by a *value* denoting kilograms or the *temperature of a stone* by a *value* denoting degrees Celsius. In other cases, such as *texture* a simple measurement unit to express this attribute on a linear scale

is not available. In these cases we can use words from the natural language to describe an attribute.

While an entity is a self-contained unit that persists over time, the attributes of a property can change as time progresses (we will discuss the concept of *time* in more detail in Section 3.)

> ***Static Attribute: An attribute of a property that does not change its value during the IoD.***

> ***Dynamic Attribute: An attribute of a property that can change its value during the IoD.***

When investigating SoSs we are primarily interested in dynamic attributes. Since a dynamic attribute, e.g., *the color of a traffic light*, changes as time progresses, a proposition about the value of a dynamic attribute is only complete, if the instant of observation, a *timestamp* is an atomic part of the proposition (the notion of a timestamp is discussed in Section 3.).

> ***Observation of an Entity: An atomic structure consisting of the name of the entity, the name of the property, the value of the property and the timestamp denoting the instant of observation.***

An observation is generated by a *sensor system* (cf. Section 4.1).

In *computer parlance*, an observation is often expressed in the form of a statement. Let us analyze the following statement:

Engine.temperature = 90

The variable name, *Engine.temperature* designates a property (temperature) of the identified entity *engine*. The value *90* specifies the value of the engine temperature. There are some elements missing from this observation: the specification of the unit for measuring the temperature and the timestamp of the observation. These missing elements are normally taken from the *context*.

> ***Context: The set of cultural circumstances, conventions or facts, and the time that surround and have a possible influence on a particular thing, construct, event, situation, system, etc. in the UoD.***

Contextual information, as it is sometimes called, is problematic in a large system which covers diverse cultural environments, since the contexts in different parts of the large systems may not be the same. Take the above example of *Engine.temperature* from Europe to the US. In Europe it is assumed that the temperature is measured in degrees *Celsius*, while in the US it is assumed that the temperature is measured in degrees *Fahrenheit*. The same statement will thus have a different meaning, depending on the context.

> ***Sphere of Control (SoC): The sphere of control of an entity during an IoD is defined by the set of entities that are under the control of the system.***

The sphere of control of a cyber system can extend beyond the boundaries of the cyber system. For example, in a cyber system that controls the traffic lights the state of the traffic light is in the SoC of the controlling computer system.

2.2 SYSTEMS

We use the definition of the term *system* introduced in the EU Project DSOS (Dependable System-of-systems IST-1999-11585 [22, p.16]:

> ***System: An entity that is capable of interacting with its environment and may be sensitive to the progression of time.***

By 'sensitive to the progression of time' we mean the system may react differently, at different points in time, to the same pattern of input activity, and this difference is due to the progression of

time. A simple example is a time-controlled heating system, where the temperature set-point depends on the current time [22, p.16]. The role of humans in a system is discussed at length below in this Section.

> ***Environment of a System: The entities and their actions in the UoD that are not part of a system but have the capability to interact with the system.***

In *classical system engineering*, the first step in the analysis of a system is the establishment of an *exact boundary* between the system under investigation and its environment.

> ***System Boundary: A dividing line between two systems or between a system and its environment.***

In SoS Engineering, such an approach can be problematic, because in many SoS the system boundary is dynamic. Consider, e.g., a *car-to-car SoS* that consists of a *plurality of cars* cruising in an area. Where is the boundary of such an SoS? A good concept should be stable, i.e., its important properties, such as size, should remain fairly constant during the IoD. The *boundary* of the *car-to-car SoS* does not satisfy this requirement and is thus a *poor concept*. Our analysis of many other existing SoSs [21], e.g., the worldwide ATM system or a smart grid system came to a similar conclusion: It is hardly possible to define a stable boundary of an SoS.

In the above example of a *car-to-car SoS* each individual car in the system (consisting of the *mechanics of the car*, the *control system within the car* and the *driver*) can be considered as an *autonomous system* that tries to achieve its given objective without any control by another system.

> ***Autonomous System: A system that can provide its services without guidance by another system.***

Before starting with the detailed design of a large system an overall blueprint that establishes the framework of the evolving artifact should be developed.

> ***System Architecture: The blueprint of a design that establishes the overall structure, the major building blocks and the interfaces between these major building blocks and the environment.***

Every organization that develops a system follows a set of explicit or implicit rules and conventions, e.g., naming conventions, representation of data (e.g, *endianness* of data), protocols etc. when designing the system.

> ***Architectural Style: The set of explicit or implicit rules and conventions that determine the structure and representation of the internals of a system, its data and protocols.***

Many of the existing legacy systems have been designed in the context of a single organization that follows its often ill-documented idiosyncratic architectural style. For example, undocumented implicit assumptions about the attributes of data can lead to mismatches when data is sent from one subsystem to another subsystem in an SoS.

> ***Monolithic System: A system is called monolithic if distinguishable services are not clearly separated in the implementation but are interwoven.***

Many systems are not monolithic wholes without any internal structure, but *are composed of interrelated parts, each of the latter being in turn hierarchic in structure until we reach some lowest level of elementary subsystem [24, p.184]*.

> ***Subsystem: A subordinate system that is a part of an encompassing system.***

The systems that form the lowest level of a considered hierarchy are called components.

> ***Component: A subsystem of a system, the internal structure of which is of no interest.***

The decomposition of a system into subsystems can be carried out until the internal structure of a subsystem is of no further interest. We call such an elementary subsystem a *component*.

Legacy System: An existing operational system within an organization that provides an indispensable service to the organization.

Homogenous System: A system where all sub-systems adhere to the same architectural style.

If a system is designed by a single organization, then all subsystems will follow the same architectural style.

Cyber-Physical System (CPS): A system consisting of a computer system (the cyber system), a controlled object (a physical system) and possibly of interacting humans.

An *interacting human* can be a prime mover or role player.

Prime mover: A human that interacts with the system according to his own goal.

An example for a prime mover could be a legitimate user or a malicious user that uses the system for his own advantage.

Role player: A human that acts according to a given script during the execution of a system and could be replaced in principle by a cyber-physical system.

A CPS is composed not only of the computer system, i.e., the cyber system, but also of a controlled object and possibly a human role player.

Entourage of a CPS: The entourage is composed of those entities of a CPS (e.g., the role playing human, controlled object) that are external to the cyber system of the CPS but are considered an integral part of the CPS.

In some CPSs, the boundary between the entourage and the environment of the CPS is dynamic, making it difficult to precisely define the scope of a CPS.

Many CPSs are Real-Time (RT) systems.

Real-time system: A computer system for which the correct results must be produced within time constraints.

Deadline: An instant when a computational action or a communication action must be completed.

We can classify deadlines according to the consequence of missing the deadline.

Hard Deadline: A deadline for a result is hard if a catastrophe can occur in case the deadline is missed.

Hard deadlines are occurring in hard real-time systems.

Firm Deadline: A deadline for a result is firm if the result has no utility after the deadline has passed.

Firm deadlines occur in Multimedia systems.

Evolutionary System: A system where the* external system boundary *is dynamic (i.e., changes during the IoD).

Our AMADEOS analysis of the state of the art in SoS engineering has shown that many fielded SoSs are *evolutionary systems*. In an evolutionary system the external system boundary is constantly changing.

The point of interaction of a system with another system or its environment is called an interface.

Interface: A point of interaction of a system with another system or with the system environment.

In some models of physics, e.g., thermodynamics, the notion of a *closed system* is introduced in order to simplify the analysis of the system.

Closed System: A system that is not interacting with its environment during a given IoD.

A closed system is a concept that does not exist in the physical world. To contrast the notion of a *closed system*, the term *open system* has been coined.

Open System: A system that is interacting with its environment during the given IoD.

2.3 SYSTEM-OF-SYSTEMS

We decided to select the following definition of Jamishidi as the starting point of our work [23].

System-of-Systems (SoS): An SoS is an integration of a finite number of constituent systems (CS) which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal.

We consider the phrase *that are networked together for a period of time* an important part of this definition, since it denotes that a static scope of an SoS may not exist and the boundary between an SoS and its environment can be dynamic.

Constituent System (CS): An autonomous subsystem of an SoS, consisting of computer systems and possibly of controlled objects and/or human role players that that interact to provide a given service.

A human who is a prime mover is considered to be a constituent system.

Dahmann and Baldwin have introduced the following four categories of SoSs [24]:

Directed SoS: An SoS with a central managed purpose and central ownership of all CSs.

An example would be the set of control systems in an unmanned rocket.

Acknowledged SoS: Independent ownership of the CSs, but cooperative agreements among the owners to an aligned purpose.

Collaborative SoS: Voluntary interactions of independent CSs to achieve a goal that is beneficial to the individual CS.

Virtual SoS: Lack of central purpose and central alignment.

While a *directed SoS,* e.g., the CSs in an automobile that are under strict central management and ownership of a car company, comes close to a homogenous system, the other extreme, a *virtual CS*, lacks the elements of homogeneity and is formed by heterogeneous subsystems belonging to very different organizations.

The CSs of an SoS exchange *information* by messages across interfaces. Any change in any of the syntactic, semantic or temporal properties of these interfaces has a potential effect on the overall operation of this SoS. We therefore call these interfaces relied upon message interfaces (RUMI).

Relied upon Message Interface (RUMI): A message interface for the exchange of information among two or more CSs that establishes a well-defined boundary between the CSs that forms part of a backbone of an SoS system architecture.

There are interfaces among the subsystems of a constituent system which are hidden behind the RUMI of the CS and which are of no relevance for the user of a RUMI. E.g., the interface between a physical sensor and the system that captures the raw data, performs data conditioning and presents the refined data at the RUMI.

The issues of *information exchange* and *interface specification* are the core topics of further Sections of this report.

3 TIME

The focus of the previous Section was on the *static structure* of systems and their parts. In this Section we start being concerned with *change*. The concept of *change* depends on the progression of *time* that is one of the core topics that is investigated in AMADEOS. In an SoS a *global notion of time* is required in order to

- Enable the interpretation of timestamps in the different CSs.
- Limit the validity of real-time control data.
- Synchronize input and output actions across nodes.
- Provide conflict-free resource allocation.
- Perform prompt error detection.
- Strengthen security protocols.

We base our model of time on *Newtonian physics* and consider time as an independent variable that progresses on a dense time-line from the past into the future. For a deep discussion on the issues of time we refer to the excellent book by Withrow, *The Natural Philosophy of Time* [25] that considers also the revision to the Newtonian model by the theory of relativity. From the perspective of an SoS the *relativistic model of time* does not bring any new insights above those of the Newtonian model of time.

Time in biology is extensively treated in the book by Winfree, *The Geometry of Biological Time* [26].

3.1 BASIC CONCEPTS

In this Section we introduce some of the basic concepts of time that are needed when discussing the temporal properties of System-of-Systems (SoSs).

Timeline: A dense line denoting the independent progression of physical time from the past to the future.

The directed time-line is often called the *arrow of time*. According to Newton, time progresses in dense (infinitesimal) fractions along the arrow of time from the past to the future.

Instant: A cut of the timeline.

There is a special instant on the timeline, the instant *now* that separates the past from the future.

Event: A happening at an instant.

An event is a happening that reports about some change at an instant.

Instants are totally ordered, while events are only partially ordered. More than one event can happen at the same instant.

Temporal order: The temporal order of events is the order of events on the time line.

If events are occurring close to each other, closer than the granularity of a digital clock, then an existing temporal order of the events cannot be established on the basis of the timestamps of the events (see the discussion on timestamp in a later part of this Section).

Causal order: A causal order among a set of events is an order that reflects the cause-effect relationships among the events.

Temporal order and causal order are related, but not identical. Temporal order of a cause event followed by an effect event is a necessary prerequisite of causal order, but causal order is more than temporal order [Kop11, p.53].

Interval: A section of the timeline between two instants.

While an interval denotes a section of the timeline between two instants, the duration informs about the length only, but not about the position of such a section.

Duration: The length of an interval.

The *length of an interval* can only be measured if a standard for the duration is available. The *physical SI second* is such an international standard (the *International System of Units* is abbreviated by *SI*).

Second: An internationally standardized time measurement unit where the duration of a second is defined as 9 192 631 770 periods of oscillation of a specified transition of the Cesium atom 133.

The physical second is the same in all three important universal time standards, UTC, TAI and GPS time. UTC (*Universal Time Coordinated*) is an astronomical time standards that is aligned with the rotation of the earth. Since the rotational speed of the earth is not constant, it has been decided to base the SI second on atomic processes establishing the International Atomic Time *TAI (Temps Atomique International).* On January 1, 1958 at 00:00:00 TAI and UTC had the same value. The TAI standard is chronoscopic and maintained as the weighted average of the time kept by over 200 atomic clocks in over 50 national laboratories. TAI is distributed world-wide by the satellite navigation system GPS (*Global Positioning System*).

The position of an instant on a standardized timeline can only be specified if a starting point for measuring the progression of time (in seconds) has been established.

Epoch: An instant on the time-line chosen as the origin for time-measurement.

GPS represents the progression of TAI time in weeks and full seconds within a week. The week count is restarted every 1024 weeks, i.e, after 19.6 years. The Epoch of the GPS signal started at 00:00:19 TAI on January 6, 1980 and again, after 1024 weeks at 00:00:19 TAI on August 22, 1999.

Cycle: A temporal sequence of significant events that, upon completion, arrives at a final state that is related to the initial state, from which the temporal sequence of significant events can be started again.

An example for a cycle is the rotation of a crankshaft in an automotive engine. Although the duration of the cycle changes, the sequence of the significant events during a cycle is always the same.

Period: A cycle marked by a constant duration between the related states at the start and the end of the cycle.

Periodic Systems are of utmost relevance in control applications

Periodic Systems: A system where the temporal behaviour is structured into a sequence of periods.

Periodicity is not mandatory, but often assumed as it leads to simpler algorithms and more stable and secure systems [29, p.19-4].

Note that the difference between *cycle* and *period* is the constant duration of the period during the IoD.

Phase: A measure that increases linearly in each period from 0 degrees at the start until 360 degrees at the end of the period.

The phase is an important concept in periodic systems that exhibit a regular behaviour and interact by the exchange of messages.

Phase alignment: The alignment of the phases between two periodic systems exhibiting the same period, such that a constant offset between the phases of the two systems is maintained.

A tight phase alignment between the computational and communication actions executed in different CSs minimizes the overall time of a transaction between a stimulus from the environment and a response to the environment.

Offset of events: The offset of two events denotes the duration between two events and the position of the second event with respect to the first event on the timeline.

3.2 CLOCKS

Time is measured with clocks. In the cyber-domain, digital clocks are used.

Clock: A (digital) clock is an autonomous system that consists of an oscillator and a register. Whenever the oscillator completes a period, an event is generated that increments the register.

The state of the register of a clock is often called the *state of clock*. The state of a clock remains constant during a complete period of the oscillator.

There exist other clocks, e.g., a *sun dial*, which is not digital in nature. The time resolution of every digital clock is limited by the duration of the period of the oscillator.

Tick: The event that increments the register is called the tick of the clock.

Granularity/Granule of a clock: The duration between two successive ticks of a clock is called the granularity of the clock or a granule of time.

The granularity of the clock can only be measured if another clock with a finer granularity is available.

We introduce a *reference clock* as a *working hypothesis* for measuring the instant of occurrence of an event of interest (such as, e.g., a clock tick) and make the following three hypothetical assumptions: (i) the reference clock has such a small granularity, e.g., a *femto second* (10^{-15} sec), that digitalization errors can be neglected as second order effects, (ii) the reference clock can observe every event of interest without any delay and (iii) the state of the reference clock is always in perfect agreement with TAI time.

Reference clock: A hypothetical clock of very fine granularity, the state of which is in agreement with TAI.

Every *good (fault-free) free-running clock* has an individual granularity that can deviate from the specified *nominal granularity* by an amount that is contained in the specification document of the physical clock under investigation.

Drift: The drift of a physical clock is a quality measure describing the frequency ratio between the physical clock and the reference clock.

Since the drift of a good clock is a number close to 1, it is conducive to introduce a drift rate by

Drift Rate = |Drift – 1|

Typical clocks have a drift rate of 10^{-4} to 10^{-8}. There exists no perfect clock with a drift rate of 0. The drift rate of a good clock will always stay in the interval contained in the specification document of the clock. If the drift rate of a clock leaves this specified interval, we say that the clock has *failed*.

Timestamp (of an event): The timestamp of an event is the state of a selected clock at the instant of event occurrence.

Note that a timestamp is always associated with a selected clock. If we use the reference clock for time-stamping, we call the time-stamp *absolute*.

Absolute Timestamp: An absolute timestamp of an event is the timestamp of this event that is generated by the reference clock.

If two events are *timestamped* by two different clocks, the temporal order of the events can be established on the basis of their timestamps only if the two clocks are synchronized.

Clock synchronization establishes a *global notion of time*. A global notion of time is required in an SoS if the timestamps generated in one CS must be interpreted in another CS. *Global time* is an abstraction of physical time in a distributed computer system. It is approximated by a properly selected subset *of the ticks* of each synchronized local clock of an ensemble. A *selected tick* of a local *clock* is called a *tick* of the *global time*. For more information on the global notion of time see [2, pp. 58-64].

> ***Precision: The precision of an ensemble of clocks denotes the maximum offset of (distance) respective ticks of the global time of any two clocks of the ensemble over the IoD. The precision is expressed in the number of ticks of the reference clock.***

The precision of an ensemble of clocks is determined by the quality of the oscillators, by the frequency of synchronization, by the type of synchronization algorithm and by the jitter of the synchronization messages. Once the precision of the ensemble has been established, the granularity of the global time followed by applying the *reasonableness condition*.

> ***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.***

We distinguish between two types of clock synchronization, internal clock synchronization and external clock synchronization.

> ***Internal Clock Synchronization: The process of mutual synchronization of an ensemble of* clocks *in order to establish a* global time *with a bounded* precision.**

There are a number of different internal synchronization algorithms, both non-fault tolerant or fault-tolerant, published in the literature (see, e.g, [28], and many others). These algorithms require the cooperation of all involved clocks.

> ***External Clock Synchronization: The synchronization of a clock with an external time base such as GPS.***

If the clocks of an ensemble are externally synchronized, they are also internally synchronized with a precision of $|2A|$, where A is the *accuracy*.

> ***Accuracy: The* accuracy of a clock *denotes the maximum offset of a given clock from the external time reference during the IoD, measured by the reference clock.***

3.3 TIME IN AN SOS

In an SoS, *external clock synchronization* is the preferred alternative to establish a global time, since the scope of an SoS is often ill defined and it is not possible to identify *a priori* all CSs that must be involved in the (internal) clock synchronization. A CS that does not share the global time established by a subset of the CSs cannot interpret the timestamps that are produced by this subset.

The preferred means of clock synchronization in an SoS is the external synchronization of the local clocks of the CSs with the standardized time signal distributed worldwide by satellite navigation systems, such as GPS, Galileo or GLONASS.

The GPS system, consisting at least of 24 active satellites transmit periodic time signals worldwide that are derived from satellite-local atomic clocks and seldom differ from each other by more than 20 nsec [29]. A GPS receiver decodes the signals and calculates, based on the offset among the signals, the position and time at the location of the GPS receiver. The *accuracy* of the GPS time is better than 100 nsec.

In a recent report from the GAO to the US Congress [30] on *GPS-Disruption—Efforts to Assess Risks to Critical Infrastructure and Coordinate Agency Action Should be Enhanced* it is pointed out that many of the large infrastructure SoSs in the US are already *using GPS time synchronization* on a wide scale and a disruption of the GPS signals could have a catastrophic effect on the infrastructure. In this report it is noted that *a global notion of time is required in nearly all infrastructure SoSs,* such as telecommunication, transportation, energy, etc. and this essential requirement has been met by gradually using more and more often the time signals provided by GPS, not considering what consequences a disruption of the time distribution, either accidental or intentional, has on the overall availability and function of the infrastructure. The report proposes to systematically monitor the risks associated with a GPS failure and to plan mitigating actions for such a case.

The periodic time signal, generated by a GPS receiver, can be used to discipline a quartz oscillator.

> ***GPSDO (Global Positioning System Disciplined Oscillator): The GPSDO synchronizes its time signals with the information received from a GPS receiver.***

With a well-synchronized GPSDO a drift rate in the order 10^{-10} can be achieved.

> ***Holdover: The duration during which the local clock can maintain the required precision of the time without any input from the GPS.***

According to [31, p.62] a good GPSDO has deviated from GPS time by less than 100 µsec during the loss of GPS input of one week. As long as the GPS is operating and its input is available, a GPSDO can provide an accuracy of the global time of better than 100 nsec. If there is the requirement that, the free running global time must not deviate by more than 1µsec, a holdover of up to one hour is achievable using a good GPSDO.

The measurement of the position of an event on the timeline or of the duration between two events by a *digital global time* must take account of two types of unavoidable errors, the *synchronization error* caused by the finite precision of the global time and the *digitalization error* caused by the discrete time base. If the *reasonableness condition* is respected, the sum of these errors will be less than *2g,* where *g* is the granularity of the global time. It follows that the true duration between two events d_{true} lies in the following interval around the observed value d_{obs}.

$$(d_{obs}-2g) < d_{true} < (d_{obs}+2g)$$

The duration between events that are temporally ordered can be smaller than the granularity of a single clock. This situation is even worse if two different globally synchronized clocks observe the two different events. It is therefore impossible to establish the true temporal order of events in case the events are closer together than *2g*. This impossibility result can give rise to *inconsistencies* about the perceived temporal order of two events in distributed system.

These inconsistencies can be avoided, if a minimum distance between events is maintained, such that the temporal order of the events, derived from the timestamps of the events that have been generated by different clocks of a system with properly synchronized clocks is always the same.

> ***Sparse Time: A time-base in a distributed computer system where the physical time is partitioned into an infinite sequence of active and passive intervals.***

The active intervals can be enumerated by the sequence of natural numbers and this number can be assigned as the timestamp of an event occurring in an active interval. In order to establish consistency all events that occur in the same active interval of a sparse time are considered to have occurred simultaneously. This procedure establishes *consistency* at the price of *faithfulness*, since the temporal order of events that are closer together than the distance between sparse events is lost.

> ***Sparse Events: Events that occur in the active interval of the sparse time.***

Events that are in the SoC of a computer system with access to a global time, e.g., the start of sending a message, can be delayed until the next active interval and thus can be forced to be sparse events.

Non-Sparse Events: Events that occur in the passive interval of the sparse time.

Events that are outside the SoC of the computer system and are in the SoC of the environment cannot be forced to occur in the active intervals of the sparse time base, and can therefore only be *non-sparse events*.

If all observers of a non-sparse event agree, by the execution of an agreement protocol, that the observed non-sparse event should be moved to the nearest active interval of the sparse time base, then the consistency of these events can be established at the price of a further reduced faithfulness.

4 DATA AND STATE

Systems-of-Systems (SoSs) come about by the transfer of *information* of one Constituent System (CS) to another CS. But what is *information*? How is *information* related to *data*? After a thorough investigation of the literature about the fundamental concepts *data* and *information* it is concluded, that these terms are not well-defined in the domain of information science—see also the paper by C. Zins who asked a number of computer scientists about their meaning associated with the terms *data—information—knowledge* and published the divergent views reported to him in [31]. In this Section we will elaborate on the concepts of *data* and *information* along the lines of reasoning expressed in [32].

4.1 DATA AND INFORMATION

Let us start by defining the fundamental concepts of *data* and *information* first:

Data: A data item is an artifact, a pattern, created for a specified purpose.

In cyber space, data is represented by a *bit-pattern*. In order to arrive at the meaning of the bit pattern, i.e., the *information* expressed by the bit pattern, we need an *explanation* that tells us how to interpret the given bit pattern.

Information: A proposition about the state of or an action in the world.

A proposition can be about *factual circumstances* or *plans*, i.e., schemes for action in the future. In any case, information belongs to the category of *constructs*, i.e., non-physical entities in world three of Popper [19]. Sometimes the phrase *semantic content* is used as a synonym for information.

Explanation: The explanation of the data establishes the links between data and already existing concepts in the mind of a human receiver or the rules for handling the data by a machine.

Since only the combination of *data* and an associated *explanation* can convey information, we form the new concept of an *Itom* that we consider the smallest unit that can carry *information*.

Itom: An Itom (Information Atom) is a tuple consisting of data and the associated explanation of the data.

The concept of an *Itom* is related to the concept of an *infon* introduced by Floridi [33]. However the properties we assign to an *Itom* are different from the properties Floridi assigns to an *infon*. The concept of an Itom does not make any assumptions about the truthfulness of the semantic content, the information, in the Itom. We thus can attribute factual information as true information (*correspondence theory of truth* [34]), misinformation (accidentally false) or disinformation (intentionally false), or just call it information if we do not know yet if it is true or false. It is often the case that only some time after data has been acquired it can be decided whether the information conveyed by the data is true or false (e.g., consider the case of a value error of a sensor)

When data is intended for a *human receiver* then the explanation must describe the data using concepts that are *familiar* to the intended human receiver. When data is intended for processing by a *machine*, the *explanation* consists of two parts, we call them *computer instructions* and *explanation of purpose [32]*.

The *computer instructions* tell the computer system how the *data bit-string* is partitioned into syntactic chunks and how the syntactic chunks have to be stored, retrieved, and processed by the computer. This part of the *explanation* can thus be considered as a *machine program* for a (virtual) computer. Such a machine program is also represented by a bit-string. We call the data bit-string *object data* and the instruction bit-string that *explains* the object data, *meta data*.

Object Data: Data that is the object of description by meta data.

Meta Data: Data that describes the meaning of object data.

A computer Itom thus contains digital *object data* and digital *meta data*. The recursion stops when the *meta data* is a sequence of well-defined machine instructions for the *destined* computer. In this case, the *design of the computer* serves as an *explanation for the meaning of the data*.

The second part of the explanation of an Itom, the *explanation of purpose,* is directed to humans who are involved in the design and operation of the computer system, since the *notion of purpose is alien to a computer system*. The *explanation of purpose* is part of the documentation of the cyber system and must be expressed in a form that is *understandable* to the human user/designer.

To facilitate the exchange of information among heterogeneous computer systems in the Internet, markup languages, such as the Extensible Markup Language XML [35], that help to explain the meaning of data have been developed. Since in XML the explanation is separated from the data, the explanation can be adopted to the context of use of the data. Markup languages provide a mechanism to support an explanation of data. In many situations the explanation of the data is taken implicitly from the context.

Variable: A tuple consisting of data and a name, where the name points to the explanation of the data.

A variable can thus be considered a basic form of an Itom.

Depending whether the data is used as an input to a system or as an output from a system we distinguish between input data and output data.

Input data: Data that is used as an input to a system.

Output data: Data that is produced by a system.

Output data is published at an interface of the system to the environment.

Raw data: The primary bit pattern that is produced by a sensor system.

The meaning of raw data—the primary bit pattern—depends on the *elementary discriminatory capabilities of the sensor system.* Take the example of the *eye* or a *camera* that produce as primary data an image consisting of a given number of pixels of varying color and intensity. The characteristics of the image depend not only on the properties of the observed entity, but also on the *purpose* that determines the position and view of the camera and the instant of taking the snapshot. The purpose has thus an effect on the characteristics of the image i.e., the *acquired afferent data.*

Refined data: Data that has been created by a purposeful process from the raw data to simplify the explanation of the data in a given context.

Refined data abstracts from those properties of the raw data that are not relevant for the given purpose (but maybe relevant for another purpose). Refined data is the result of the process of *feature extraction* taking the raw date as input and looking at those features in the raw data that are relevant for the given purpose. By drastically reducing the size of (raw) data, refined data, sometimes called *meaningful data*, only retains those attributes of an observation that are considered relevant from the perspective of the user purpose. Those features that are not considered relevant in the process of feature extraction are not retained in the refined data (although they are present in the *raw data*).

Whereas the representation of the *raw data* is determined by the design of the sensor system, the representation of *refined data* depends on conventions that are determined by the context where the data is used.

When data is moved from one CS to another CS of an SoS, the context may change, implying that an explanation that is context-dependent changes as well. Take the example of *temperature* expressed as a *number*. In one context (e.g., Europe) the number is interpreted as degrees Celsius, while in another context (e.g, the US) the number is interpreted as degrees Fahrenheit. If

we do not change the number (the data) then the meaning of the Itom is changed when moving the data from one context to another context. The neglected context sensitivity of data has caused accidents in SoSs [36].

An observation of a dynamic entity is only complete if the instant, the *timestamp* of making the observation, is recorded as part of the explanation.

The timestamp of input data is determined by the termination instant of the sensing process.

No timestamp is needed in the explanation when the properties of the observed entity are static. In the context of our work, we are mostly interested in dynamic properties of systems.

A periodic system often produces a stream of related data elements.

Data Frame: A valued data structure produced by a periodic system.

In a periodic system, the data contained in a given data frame is closely related to the data contained in the following data frame.

Data stream: A sequence of data frames produced by a periodic system.

The concept of data streams is important in most control systems. In many cases it is possible to mitigate the loss of a data frame by replacing it by the contents of the previous frame.

Encrypted data (called cipher-text): Data that has been subjected to a transformation (encryption) such that an unauthorized user cannot easily interpret the encrypted data.

4.2 STATE

Many systems store information about their interactions with the environment (since the start of a system with a clean memory) and use this information to influence their future behaviour.

State: The state of a system at a given instant is the totality of the information from the past that can have an influence on the future behaviour of a system.

A state is thus a valued data structure that characterizes the condition of a system at an instant. The concept of state is meaningless without a concept of time, since the distinction between past and future is only possible if the system is time-aware.

Stateless System: A system that does not contain state at a considered level of abstraction.

Statefull System: A system that contains state at a considered level of abstraction.

The variables that hold the stored state in a statefull system are called *state variables*.

State Variable: A variable that holds information about the state.

State Space: The state space of a system is formed by the totality of all state variables at that instant.

If we observe the progress of a system, we will recognize that the size of the state space grows or shrinks as time passes. The state space has a relative minimum at the start and end of an *atomic action* (see the Section 5).

The size of the state space is important if we want to restart a system after a failure (e.g., a corruption of the state by a transient fault). We have to repair the corrupted state before we can reuse the system. Generally, the smaller the state space, the easier it is to repair and restart a system.

Most control systems are cyclic or even periodic systems.

Ground State: At a given level of abstraction, the ground state of a cyclic system is a state at an instant when the size of the state space is at a minimum relative to the sizes of the state space at all other instants of the cycle.

We call the instant during the cycle of a cyclic system where the size of the state has a minimum the *ground state instant*.

Ground State Instant: The instant of the ground state in a cyclic system.

At the ground state instant all information of the past that is considered relevant for the future behaviour should be contained in a declared ground state data structure. At the ground state instant no *task* may be active and all communication channels are flushed. Ground state instants are ideal for reintegrating components that have failed.

Declared Ground State: A declared data structure that contains the relevant ground state of a given application at the ground state instant.

The *declared ground state* is essential for system recovery. The declared ground state contains only of those ground state variables that are considered *relevant* by the designer for the future operation of the system in the given application. Other ground state variables are considered non-relevant because they have only a minor influence on the future operation of the system. The decision of whether an identified state variable is relevant or not relevant depends on a deep understanding of the dynamics of an application.

Concise State: The state of a system is considered concise if the size of the declared ground state is at most in the same order of magnitude as the size of the system's largest input message.

Many control systems have a concise state. There are other systems, such as data base systems that do not have a concise state—the size of the state of a data-base system can be Gigabytes.

In contrast to state variables that hold information about the state at an instant an event variable holds information about a *change* at an instant.

Event Variable: A variable that holds information about some change of state at an instant.

5 ACTIONS AND BEHAVIOUR

We can observe the dynamics of a system that consists of discrete variables by an *event-based view* or by a *state-based view*.

In the *event-based view* we observe the state of relevant state variables at the beginning of the observation and then record all events (i.e. changes of the state variables) and the time of occurrence of the events in a trace. We can reconstruct the value of all state variables at any past instant of interest by the recorded trace. However, if the number of events that can happen is not bounded, the amount of data generated by the event-based view cannot be bounded.

In the *periodic state-based view* (called *sampling*), we observe the values of relevant state variables at selected *observation instants* (the sampling points) and record these values of the state variables in a trace. The duration between two observation instants puts a limit on the amount of data generated by the state-based view. However, the price for this limit is the loss of fidelity in our trace. Events that happen within a duration that is shorter than the duration between the equidistant observation instants may get lost.

> ***Sampling: The observation of the value of relevant state variables at selected observation instants.***

Most control systems use *sampling* to acquire information about the controlled object. The choice of the *duration between two observation instants*, called *the sampling interval,* is critical for acquiring a satisfying image of the controlled object. This issue is discussed extensively in the literature about control engineering [47].

5.1 ACTIONS

In the following we introduce some concepts that allow us to describe the dynamics of a computer system.

> ***Action: The execution of a program by a computer or a protocol by a communication system.***

An action is started at a specified instant by a *start signal* and terminates at a specified instant with the production of an *end signal*.

> ***Start signal: An event that causes the start of an action.***
>
> ***End signal: An event that is produced by the termination of an action.***

Between the start signal and end signal an action is active.

> ***Execution Time: The duration it takes to execute a specific action on a given computer.***

The execution time is data dependent.

> ***Worst Case Execution Time (WCET): The worst case data independent execution time required to execute an action on a given computer.***

There are two possible sources for a start signal of an action.

> ***Time-triggered (TT) Action: An action where the start signal is derived from the progression of time.***

An action can also be started by the completion of the previous action or by some other event (e.g., the push of a start button).

> ***Event-triggered (ET) Action: An action where the start signal is derived from an event other than the progression of time.***

We distinguish also between computational actions and communication actions.

Computational Action: An action that is characterized by the execution of a program by a machine.

Communication Action: An action that is characterized by the execution of a communication protocol by a communication system.

Communication actions will be discussed in more detail in Section 6.

In our model of an action we assume that at the start event an action reads input data and state. At the end event an action produces output data and a new state.

An action *reads* input data if the input data is still available after the action. An action *consumes* input data if the input data record is unavailable after the consumption by an action.

An action *writes* output data, if an old version of the output data is *overwritten* by the output data generated by the action. An action *produces* output data if a *new unit* of output data is generated by the action.

Input Action: An action that reads or consumes input data at an interface.

Output Action: An action that writes or produces output data at an interface.

We distinguish between actions that access the state of a system and those that do not.

Stateless Action: An action that produces output on the basis of input only and does not read, consume, write or produce state.

Statefull action: An action that reads, consumes, writes or produces state.

An action starts at the *start signal* and terminates by producing an *end signal*. In the interval <*start signal, end signal*> an action is *active*. While an action is active, the notion of state is undefined.

We can compose actions to form action sequences.

Action Sequence: A sequence of actions, where the end-signal of a preceding action acts as the start signal of a following action.

An *action sequence* is often called a *process*.

Activity Interval: The interval between the start signal and the end signal of an action or a sequence of related actions.

An action at a given level of abstraction, e.g., the execution of a program, can be decomposed into sub-actions. The decomposition ends when the internal behaviour of a sub-action is of no concern.

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.

Atomic actions are related to the notion of a component introduced above: neither the internals of components (from the point of view of structure) nor the internals of an atomic action (from the point of view of behaviour) are of interest.

Irrevocable Action: An action that cannot be undone.

An irrevocable action has a lasting effect on the environment of a system. For example consider an output action that triggers an airbag in a car.

Idempotent Action: An action is idempotent if the effect of executing it more than once has the same effect as of executing it only once.

For example, the action *move the door to 45 degrees* is idempotent, while the action *move the door by five degrees* is not idempotent. Idempotent actions are of importance in the process of recovery after a failure.

We can combine computational action and communication actions to form a transaction.

> ***Transaction: A related sequence of computational actions and communication actions.***

In a control system the duration of the transaction that starts with the observation of the controlled object and terminates with the output of the result to an actuating device has an effect on the quality of control.

> ***Transaction Activity Interval: The interval between the start signal and the end signal of a transaction.***

5.2 BEHAVIOUR

The behaviour of a system—the observable traces of activity at the system interfaces—is of utmost interest to a user.

> ***Behaviour: The timed sequence of the effects of input and output actions that can be observed at an interface of a system.***

The effect of a *consuming input action* is the consumption of the input data record, while the effect of a reading input action does not change the input data and therefore has no observable effect. This is not true at the output side. Both, a *writing output action* and a *producing output action* have an observable effect.

> ***Deterministic Behaviour: A system behaves deterministically if, given an initial state at a defined instant and a set of future timed inputs, the future states, the values and instants of all future outputs are entailed.***

A system may exhibit an intended behaviour or it may demonstrate a behaviour that is unintended (e.g. erroneous behaviour).

> ***Service: The intended behaviour of a system.***

> ***Function: The intended behaviour of a stateless system.***

The service specification must specify the intended behaviour of a system. In non real-time systems, the service specification focuses on the data aspect of the behaviour. In real-time systems the *precise temporal specification* of the service is an integral part of the specification.

6 COMMUNICATION

It is the basic objective of a communication system to transport a message from a sender to one or more receivers *within a given duration* and with a *high dependability*. By *high dependability* we mean that by the end of a specified time window the message should have arrived at the receivers with a high probability, the message is not corrupted, either by unintentional or intentional means, and that the security of the message (confidentiality, integrity, etc.) has not been compromised. In some environments e.g., the Internet of Things (IoT), there are other constraints on the message transport, such as, e.g., minimal energy consumption.

In an SoS the communication among the CSs by the exchange of messages is the core mechanism that realizes the integration of the CSs. It is imperative to elaborate on the concepts related to the message exchange with great care.

Since communication requires that diverse senders and receivers agree on the rules of the game, all involved partners must share these rules and their interpretation.

Communication Protocol: The set of rules that govern a communication action.

In the past fifty years hundreds of different communication protocols, that often differ in minor aspects only, have been developed. Many of them are still in use in legacy systems. This diversity of protocols hinders the realization of the economies of scale by the semiconductor industry.

We distinguish between two classes of protocols: basic transport protocols and higher-level protocols. The basic transport protocols are concerned with the transport of a message from a sender to one or more receivers. The higher-level protocols build on these basic transport protocols to provide more sophisticated services.

6.1 MESSAGES

Message: A data structure that is formed for the purpose of the timely exchange of information among computer systems.

We have introduced the word *timely* in this definition to highlight that a message combines concerns of the value domain and of the temporal domain in a single unit.

In the temporal domain, two important instants must be considered.

Send Instant: The instant when the first bit of a message leaves the sender.

Arrival Instant: The instant when the first bit of a message arrives at the receiver.

Receive Instant: The instant when the last bit of a message arrives at the receiver.

Transport Duration: The duration between the send instant and the receive instant.

Messages can be classified by the strictness of the temporal requirements.

In real-time communication systems strict deadlines that limit the transport duration must be met by the communication system.

From the point of view of the value domain, a message normally consists of three fields: a *header*, a *data field,* and a *trailer*. The header contains transport information that is relevant for the transport of the message by the communication system, such as the delivery address, priority information etc. The data field contains the payload of a message that, from the point of view of transport, is an *unstructured bit vector*. The trailer contains redundant information that allows the receiver to check whether the bit vector in the message has been corrupted during transport. Since a corrupted message is discarded, we can assume (as the fault model on a higher level) that the communication system delivers either a correct message or no message at all.

Payload of a Message: The bit pattern carried in the data field of the message.

The payload of a message consists of *data* that require an *explanation* to retrieve the information that is conveyed by the message. In many cases, the explanation is influenced by the context, as detailed in Section 4.1. In an SoS, where the context of the sender and the receiver are different, care must be taken that the *data* and the *explanation* are consistent in order to ensure that the information carried by the message is not distorted.

Apart from the transport information that is contained in the header of a message, the explanation of a message consists of two parts: the syntactic specification and the semantic specification.

> ***Syntactic Specification: The specification that explains how the data field of a message is structured into syntactic units and assigns names to these syntactic units.***

The names designate the *concepts* that explain the meaning of the data. An Interface description language (IDL), such as the OMG IDL [37] can be used to express the syntactic specification

> ***Semantic Specification: The specification that explains the meaning of the named syntactic units.***

There are different means to establish the link between a message and the *explanation of the message*:

- *Message name*: The payload contains an a *priori agreed* message name in a defined position. The message name points to the explanation of the message.
- *Port name*: The explanation is linked to the port where the message arrives.
- *Receive instant*: The explanation is linked to the receive instant when a periodic message arrives.
- *Self-contained message:* The explanation is at an a priori specified position of the payload of the message.

In order that errors in a message can be detected, *assertions* can be associated with a message

> ***Message assertion: A message assertion is a predicate on the values of a message and relevant state variables that defines an application specific acceptance criterion.***

Using such assertions messages can be classified as depicted in Table 2. This classification has been taken from the DSOS project ([20], p19).

Table 2: Message Classification

Attribute	Description	Antonym
Valid	A message is valid if its checksum and contents are in agreement.	Invalid
Checked	A message is checked at the source (or, in short, checked) if it passes the output assertion.	Not Checked
Permitted	A message is permitted with respect to a receiver if it passes the input assertion of that receiver. The input assertion should verify, at least, that the message is valid.	Not Permitted
Timely	A message is timely if it is in agreement with the temporal specification.	Untimely
Value correct	A message is value-correct if it is in agreement with the value specification.	Not value correct
Correct	A message is correct if it is both timely and value correct.	Incorrect
Insidious	A message is insidious if it is permitted but incorrect.	Not insidious

6.2 BASIC TRANSPORT SERVICE

A basic transport service transfers a message from a sender to one or more receivers. We limit our discussion to three different transport protocol classes that are representative for a number of important protocols in each class:

- Datagram
- PAR Message
- TT Message

Datagram: A best effort message transport service for the transmission of sporadic messages from a sender to one or many receivers.

A datagram is a very simple transport service. A datagram is forwarded along the best available route from a sender to one or a number of receivers. Every datagram is considered independent from any other datagram. It follows that a sequence of datagrams can be reordered by the communication system. If a datagram is corrupted or lost, no error will be indicated.

PAR-Message: A PAR-Message (Positive Acknowledgment or Retransmission) is an error controlled transport service for the transmission of sporadic messages from a sender to a single receiver.

In *the positive acknowledgment-or-retransmission* (PAR) protocol a sender waits for a given time until it has received a positive acknowledgement message from the receiver indicating that the previous message has arrived correctly. In case the timeout elapses before the acknowledgement message arrives at the sender, the original message is retransmitted. This procedure is repeated n-times (protocol specific) before a permanent failure of the communication is reported to the high-level sender. The jitter of the PAR protocol is substantial, since in most cases the first try will be successful, while in a few cases the message will arrive after *n* times the timeout value plus the worst-case message transport latency. Since the timeout value must be longer than two worst-case message transport latencies (one for the original message and one for the acknowledgment) the jitter of PAR is longer than **(*2n*)** worst-case message-transport latencies [2, p.169]. In addition to this basic PAR protocol one can find many protocol variants that refine this basic PAR protocol.

TT-Message: A TT-Message (Time-Triggered) is an error controlled transport service for the transmission of periodic messages from a sender to many receivers.

A time-triggered message is a periodic message that is transported from one sender to one or more receivers according to a pre-planned schedule. Since it is known *a priori* at the sender, the receiver and the communication system when a time-triggered message is expected to arrive, it is possible to avoid conflicts and realize a tight phase alignment (see Section 3.1) between an incoming and an outgoing message in a communication system switch. The error detection of a TT-message is performed by the receiver on the basis of his a priori knowledge about the expected arrival time of a message. The error detection latency is determined by the precision of the global time.

We will investigate these three basic transport protocols from the following points of view

- Data/Control Flow and Error Detection
- Message Handling
- Transport Duration

and present a summary of our investigation at the end of this Section.

6.2.1 Data/Control Flow and Error Detection

A message is both, a container for delivering a payload and a carrier of control signals.

Data flow: The flow of the payload data of a message from a sender to the receivers.

The *data flow view* only looks at the flow of payload data and is not concerned with the control signals associated with a message transmission.

Control flow: The flow of control signals when executing a protocol.

In the datagram and TT messages, *data flow* and *control flow* are uni-directional. In a PAR message, the data flow is *uni-directional* while the control flow is *bi-directional*. The bi-directional control flow in the PAR message is needed to implement flow control and error detection.

Flow control: The control of the flow of messages from the sender to the receiver such that the sender does not outpace the receiver.

We distinguish between two types of flow control, explicit flow control and implicit flow control.

Explicit flow control: After having sent a message, the sender receives a control message from the receiver informing the sender that the receiver has processed the sent message.

In a PAR message, the sender uses the explicit flow control message also to perform err*or detection by the sender*. By associating a time-out with the instant of arrival of the explicit control message the sender can detect a communication error. The error detection latency in PAR is relatively long, since an error will only be reported to a client after all retries to send the message have failed.

Implicit flow control: The sender and receiver agree a priori on a maximum send rate. The sender commits to never send messages faster than the agreed send rate and the receiver commits to accept all messages that the sender has sent.

The TT message uses implicit flow control based on the a priori knowledge of the send schedule by the sender and receiver. The *error detection* of a TT-message is performed by the *receiver* based on his a priori knowledge of the expected arrival times of the periodic message. The error detection latency of a TT message is determined by the precision of the global time.

6.2.2 Message Handling

Communication systems differ in the way they handle the messages at the sender and at the receiver. We distinguish between the *consume/produce* and *read/write* paradigm for handling messages at the sender and receiver.

Consume/Produce (CP) paradigm: At the sender, the communication system consumes the message from a sender queue and at the receiver the communication system adds the received message to a receiver queue.

The CP paradigm requires a queue management at the sender and at the receiver and must deal with all cases of *queue emptiness* and *queue overflow*. This is called back-pressure flow control.

Back-pressure flow control: A message control schema, where the receiver exerts back pressure on the sender to ensure that the speed of the sender will not outpace the speed of the receiver and thus lead to queue overflow.

Read/Write (RW) paradigm: At the sender the communication system reads the contents of the message from a message variable and at the receiver the communication system writes the arriving message into a message variable, overwriting the old content of the message variable.

In the RW paradigm no queue management and back-pressure flow control is required but a message that is not read by the receiving process before the next message arrives is lost. In real-time systems, the RW transport schema, that always presents the most recent version of a message to the receiving process, is widely used.

6.2.3 Transport Duration and Jitter

In real-time systems the duration and the jitter of a message from a sender to a receiver is of utmost importance.

> ***Jitter: The duration between the minimal transport duration and the maximum transport duration.***

While the transport duration of a datagram is a priori unknown, the transport duration of a PAR message is bounded, but the jitter of a PAR message might be significant.

A TT message has a bounded transport duration and a small jitter that is determined by the precision of the clock synchronization.

6.2.4 Summary

Table 3 presents the summary of our analysis.

Table 3: Characteristics of Basic Transport Services

Characteristic	Datagram	PAR-Message	TT- Message
Send Instants	Sporadic	Sporadic	Periodic
Data/Control Flow	Uni-directional	Bi-directional	Uni-directional
Flow Control	None	Explicit	Implicit
Message Handling	R/W or C/P	C/P	R/W
Transport Duration	a priori unknown	Upper limit known	Tight limit known
Jitter	Unknown	large	small
Temporal Error Detection	None	At Sender	At Receiver
Example	UDP	TCP/IP	TT-Ethernet

Although the basic datagram service does not provide temporal error detection, a posteriori error detection of datagram messages can be achieved by putting the send timestamp in the message, given that synchronized clocks are available.

It is up to the application to decide which basic transport protocol is most appropriate to integrate the CSs into an SoS.

6.3 HIGH-LEVEL PROTOCOLS

The transport protocols form the basis for the design of higher-level protocols, such as protocols for *file transmission* and *file sharing*, *device detection* and numerous other tasks. It is beyond the scope of this conceptual model to discuss the diversity of higher-level protocols. In the latter part of the AMADEOS project we will look at some of the higher level protocols that are of particular relevance in SoSs.

However, it must be considered that the temporal properties of the basic transport protocol determine to a significant extent the temporal properties of the high-level protocols.

7 EMERGENCE

System-of-Systems (SoS) are built to realize novel services that go beyond the services that can be provided by any of the CSs in isolation. We call these novel services *emergent services.* The concept of emergence is thus at the core of SoS engineering and warrants an in-depth analysis of the topic.

Take the example of the worldwide ATM system. A local ATM system gives a client access to his financial assets via an ATM terminal in the form of the local currency at the home base. The worldwide ATM system realizes a novel service: the access of financial assets of a client from any ATM terminal in the world in a currency that is determined by the local context of the foreign ATM terminal. This novel emergent service cannot be provided by any ATM system in isolation but comes about by the worldwide integration of all ATM terminals. Incidentally, this example is also a good illustration of the concept of an *itom* introduced in Section 4.1. The *same value* of a financial transaction is represented in different countries (contexts) by the local currency of the country, implying that the data (the amount of money in the local currency) and the representation (the local currency) are different, while the *semantic content*, the value of the transaction remains the same.

The study of the topic of emergence, how novel phenomena emanate into the world by the interaction and integration of lower-level components, has been an important subject of research in the domains of the natural sciences and philosophy in the long history of the sciences. However, we could not find a generally accepted explanation of the concept of emergence, although there are some excellent publications devoted to this topic, e.g. [40, 41, 42, 43, 44]. We align the concepts of emergence introduced in the AMADEOS conceptual model of an SoS with the concepts published in the general literature about emergence.

7.1 DEFINITION OF EMERGENCE

In his seminal paper *The Architecture of Complexity* H. Simon discusses the central role that hierarchies play in models of complex systems [24, p. 217]. Related primitive elements at the bottom of a hierarchy are clustered together to form a *whole* for the next higher level. This process of agglomeration is repeated recursively until the top of the hierarchy the *apex*, is reached. The apex is determined by the purpose of the model.

Simon distinguishes between two different types of hierarchies, a *formal hierarchy* and a *non-formal hierarchy* or, often called, a *holarchy [43]*. In a formal hierarchy each subsystem at level *n* is linked vertically to its controlling system at level *n+1*, the level above, by a *reporting and control relation*. *More exactly, in a hierarchic formal organization, each system consists of a "boss" and a set of subordinate subsystems* [24, p.196]. In a formal hierarchy there are no direct horizontal interactions among the subsystems of a given level. An example for a formal hierarchy is the command structure of a military organization.

In a *non-formal hierarchy* or *holarchy,* there are *horizontal interactions* among the related sub-systems at level *n* that lead to the formation of a *whole* with its own characteristic properties at the level above, level *n+1*. Koestler [45, p.341] introduces the notion of a *holon*, a two-faced subsystem as the central concept in a non-formal hierarchy.

> ***Holon: A two-faced entity in a non-formal hierarchy that externally (upwards and horizontally) acts as a whole but internally (downwards) is established by the interactions of its parts (i.e., the interactions among the holons of the level below).***

According to Koestler holons form the basic building blocks of holarchies.

> ***Holarchy: A structure where holons at one level interact horizontally to form a novel holon at the next higher level.***

An example of a holarchy is the hierarchic model of the universe, starting from subatomic elements up to the appearance of the human species, where higher-level subsystems emerge out of the interaction and organization of the primitive lower level physical entities.

It is generally agreed that a definition of emergence must focus on two adjacent levels of a hierarchic system. The lower level, called the *micro-level,* is often called the *level of the parts* and the higher level, the *macro-level*, is the *level of the whole* where the novel emergent phenomena appear. In an SoS context, the micro-level is the level of the CSs, while the macro-level is the level of the SoS. We introduce the following definition of emergence taken from [32]:

> ***Emergence: A phenomenon of a whole at the macro-level is emergent if and only if it is new with respect to the non-relational phenomena of any of its proper parts at the micro level.***
>
> ***Resultant phenomenon: A phenomenon at the macro-level is resultant if it can be reduced to a sum of phenomena at the micro-level.***

The essence for the occurrence of emergent phenomena at the macro-level lies in the *organization of the parts*, i.e., in the spatial arrangement and/or the physical or informational interactions among the parts at the micro-level. Although this definition requires that the emergent phenomenon does not appear at any level below the macro (SoS) level, i.e., emergent phenomena are *systemic,* the definition is extensive and includes many phenomena that are familiar to us from our every-day experience.

According to the above definition of emergence, the emergent phenomenon must be *new* with respect to the non-relational phenomena of the parts, but not with respect to the *conceptual landscape* of the observer. There are some publications that relate the appearance of emergence to a *surprise* by the observer. Such a view of emergence would mean that the concept is relative to the epistemic state of the observer. One observer, who knows about this phenomenon is *not surprised*, and another one, who sees the phenomenon for the first time is *surprised*. We do not feel that such a relative definition of the concept of emergence is useful.

It is often the case that novel concepts have to be formed to capture the emergent phenomena at the macro level and new laws, called *intra-ordinal laws [44]*, have to be introduced to be able to describe the emerging phenomena at the macro level appropriately.

> ***Intra-ordinal Law: A new law that deals with the emerging phenomena at the macro level.***

These phenomena and laws are either absent at the level below or are simply meaningless at a lower level [47, p.36]. Intra-ordinal laws of an *axiomatic nature* have an explanatory power for phenomena at *all levels above* the macro level for which they have been introduced. Whether a newly formulated intra-ordinal law is of an axiomatic nature, i.e. is a universally accepted principle, is a topic of intense philosophical debates.

For example, if we consider a plurality of water molecules under appropriate environmental conditions *fluidity* and *wetness* are new concepts that are introduced to describe the emergent phenomena of the *compound physical entity* water. The *intra-ordinal Navier-Stokes law* explains and predicts the *fluid dynamics* of *water*—this law is meaningless if there is no state of liquidity. In contrast, the weight of water is *resultant*.

A second category of laws, called trans-ordinal laws [44], explains the appearance of the emergent phenomena.

> ***Trans-ordinal Law: A Law that explains the emergence of the whole and the new phenomena at the macro-level out of the properties and interactions of the parts at the lower adjacent micro-level.***

We distinguish between two types of emergence, *weak emergence* and *strong emergence*. This classification can only be accomplished after a thorough analysis of the emergent phenomenon has been carried out.

> ***Weak emergence: An emergent phenomenon that is observed at a macro level is weakly emergent if a trans-ordinal law that explains the occurrence of the emergent phenomenon at the macro level out of the properties and interactions of the parts at the adjacent micro level is known (or has been formulated post facto).***
>
> ***Strong Emergence: An emergent phenomenon that is observed at the macro level is strongly emergent if, after a careful analysis of the emergent phenomenon, no trans-ordinal law that explains the appearance of the emergent phenomenon at the macro level out of the properties and interactions of the parts at the adjacent micro level is known (at least at present).***

The distinction between *weakly emergent* and *strongly emergent* depends on the *state of knowledge in the scientific community.* If, after a phase of detailed observation and analysis, an emergent phenomenon cannot be explained on the basis of existing knowledge about the constituent systems and their interactions, then it is classified as *strongly emergent.* Often the extension or modification of known laws at the micro-level will provide for some later *post facto* explanation of the emergent phenomenon. The emergent phenomenon will then be reclassified as *weakly emergent*.

There are some strongly emergent phenomena that up to now cannot be explained, based on the present state of knowledge, although an immense effort has been invested in order to gain an understanding of these phenomena. Examples for these phenomena are the *emergence of life* or the *emergence of the mind*. It is an ongoing philosophical debate whether *our limited knowledge* or *ontological novelty* is the reason for the difficulty in explaining these strongly emergent phenomena.

The principle of *supervenience* [46] establishes an important dependence relation between the emerging phenomena at the macro-level and the interactions and arrangement of the parts at the micro-level.

> ***Supervenience: The principle of Supervenience states that (Sup i) a given emerging phenomenon at the macro level can emerge out of many different arrangements or interactions of the parts at the micro-level while (Sup ii) a difference in the emerging phenomena at the macro level requires a difference in the arrangements or the interactions of the parts at the micro level.***

Because of *Sup-i* one can abstract from the many different arrangements or interactions of the parts at the micro level that lead to the same emerging phenomena at the macro level. *Sup-i* entails a *significant simplification* of the higher-level models of a holarchy.

Because of *Sup-ii* any difference in the emerging phenomena at the macro level can be traced to some significant difference at the micro level. *Sup-ii* is important from the point of view of *failure analysis*.

Let us look at the example of a transistor. The *transistor effect* is an emergent effect caused by the proper arrangement of dopant atoms in a semiconducting crystal. The exact arrangement of the dopant atoms is of no significance as long as the provided behavioural specifications of a transistor are met. In a VLSI chip that contains millions of transistors, the detailed microstructure of every single transistor is probably unique, but the external behaviour of the transistors (the holons) is considered the *same* if the behavioural parameters are within the given specifications. It is a tremendous simplification for the designer of an electronic circuit that she/he does not have to consider the unique microstructure of every single transistor.

In the literature about emergence the topic of *downward causation* of the emergent phenomena is intensively discussed.

Downward Causation: The phenomenon that some novel higher-level properties cannot be reduced to mere physical phenomena and have causal powers to control the lower-level process from which they emerge.

There seems to be no universal agreement on the explanation of the phenomenon of downward causation.

7.2 CLASSIFICATION OF EMERGENCE IN AN SOS

An SoS is formed by the information interactions and physical interactions among *Constituent Systems (CSs)*, e.g,. new or existing monolithic systems (called *legacy systems*). When analyzing emergence in an SoS, the focus is on two distinct adjacent levels: the micro-level of the CSs that interact via messages to cause the emergent phenomena (behaviour) of interest at the macro-level, the level of the SoS.

The type of emergence that is occurring in the cyber part of an SoS is *weak emergence,* even if we are surprised and cannot explain the occurrence of an unexpected emergent phenomenon at the moment of its first encounter. If we have made proper provisions to observe and document all interactions (messages) among the CSs in the domains of time and value we can replay and analyze the scenario after the fact. At the end, we will find the mechanisms that explain the occurrence of the emergent phenomenon. There is no ontological novelty in the interactions of the CSs in the cyber parts of an SoS.

Table 4 depicts four cases of emergent behaviour that must be distinguished in an SoS. *Expected and beneficial* emergent behaviour is the normal case. Un*expected and beneficial emergent* behaviour is a positive surprise. *Expected detrimental emergent* behaviour can be avoided by adhering to proper design rules. The problematic case, which we investigate in the following in more detail, is *unexpected detrimental emergent behaviour*.

Table 4: Classification of Emergent Behaviour

Contribution of the emergent behaviour to the overall goal of a system **Emergent Behaviour**	**Beneficial**	**Detrimental**
Expected	**Normal case**	**Avoided by appropriate rules**
Unexpected	**Positive surprise**	**Problematic case**

8 PROBLEM SOLVING

This Section will be expanded in the final version of the conceptual model.

8.1 BASIC CONCEPTS

Many systems are developed in order to provide a service that helps a user in the solution of some identified problem.

Problem: A perceived need to transform an initial state to a goal state.

The initial state, the starting state for a problem solving activity, can be classified as follows:

Initial State: (i) an existing deficient state of affairs that needs a solution or (ii) a recognized opportunity that should be exploited or (iii) a formal statement of a question (academic story problem).

In a real-life problem scenario, there are normally many states that can be considered a solution to a given problem.

Solution Set: The set of states that are considered a solution to the problem at hand.

The goal state is any element of the solution set.

Goal State: An element of the solution set.

In the first phase, a problem solver must develop a decision procedure to determine whether an element of the state space is a member of the solution set.

Problem Space: A state space that contains the initial state, the solution states and the identified constraints that the paths form the initial state to the solution state must satisfy.

Acceptance Test: A test that determines if a state in the problem space is a member of the solution set.

Normally, the size of the problem space is limited by a set of given constraints.

Constraint: A restriction in the problem space.

We call a possible path that leads from the initial state through the problem space to a goal state a solution path.

Solution Path/Plan: A path of intermediate states from the initial state to the goal state, considering the given constraints.

Some states of a solution path are important intermediate results.

Sub Goal: A significant intermediate state on the solution path.

To summarize, we consider the following concepts as the problem elements:

Problem Elements: The initial state, the solution states, the acceptance test, the sub goals and the constraints that must be considered on the solution path.

8.2 PROBLEM TYPES

It is useful to distinguish between the structural and situational characteristics of a problem.

Structural Characteristics: Representation of the abstract structure of the problem, focusing on the problem elements and their interactions.

The situational characteristics focus on the context of the problem.

Situational Characteristics: Representation of the context of the problem that anchors the problem in the real world and elaborates the constraints.

Problems can be classified along a scale from well-structured to ill-structured.

Well-structured Problem: A problem, where the initial state, the solution states, the acceptance test, and the constraints are well defined.

Well-structured problems are dominated by the structural characteristics. Most textbook problems are well-structured problems.

Ill-structured Problem: A problem where either the initial state and/or the solution states, and/or the acceptance test and/or constraints, are not well defined.

Ill-structured problems are dominated by the situational characteristics. Most real-life problems are ill-structured problems.

Problem Framing: The process of describing and interpreting a problem within the problem domain (also the point of view, from which the problem is described).

We can distinguish between the following problem types (from *well-structured* to *ill-structured*):

Formal Problem: A problem in a well-defined problem space.

Example: Textbook problem, logic problems, algorithmic problems.

Decision Problem: A problem to select an alternative out of a set of given alternatives.

Example: Should I accept an offered job?

Diagnosis Problem: A problem to find the cause of observed symptoms.

Example: Medical Diagnosis, technical troubleshooting.

Strategy Problem: A problem to devise a strategy or conceive a design that satisfies an abstract goal state under a myriad of explicit and implicit constraints.

Example: devise a policy to reduce the deficit.

Problem Solving: The process of finding a satisficing goal state. Problem solving encompasses (i) Analyzing and specifying the initial state (ii) Exploring the constraints in the problem domain (iii) Finding one or a set of satisficing goal states and (iv) Finding a solution path from the initial state to a selected goal state that observes the given constraints.

In the next phase of the AMADEOS project we plan to investigate which phases of the problem solving process can be supported by an SoS and by what means an SoS can contribute to problem solving.

9 DEPENDABILITY AND SECURITY

In this Section we define the basic concepts related to dependability and security. Dependability and security are important properties for System-of-Systems (SoSs) since they impact availability/continuity of operations, reliability, maintainability, safety, data integrity, data privacy and confidentiality. Definitions for dependability and security concepts can be very subtle; slight changes in wording can change the entire meaning. Thus, we have chosen to refer to basic concepts and definitions that are widely used in the dependability and security community.

The reference taxonomy for the basic concepts of dependability applied to computer-based systems can be found in [3]. [3] is the result of a work originated in 1980, when a joint committee on "Fundamental Concepts and Terminology" was formed by the TC on Fault–Tolerant Computing of the IEEE CS1 and the IFIP WG 10.4 "Dependable Computing and Fault Tolerance" with the intent of merging the distinct but convergent paths of the dependability and security communities.

In addition to the work of Laprie [3] [4], we also refer to definitions from the *CNSS Instruction No. 4009: National Information Assurance (IA) Glossary* [1]. The CNSSI 4009 was created through a working group with the objective to create a standard glossary of security terms to be used across the U.S. Government, and this glossary is periodically updated with new terms. We also cite security terms as defined by Ross Anderson's "Security Engineering: A Guide to Building Dependable Distributed Systems" [7] and Bruce Schneier's "Applied Cryptography" [8].

9.1 THREATS: FAULTS, ERRORS, AND FAILURES

The threats that may affect a system during its entire life from a dependability viewpoint are failures, errors and faults. Failures, errors and faults are defined in the following, together with other related concepts.

> ***Failure: The actual system behaviour deviation from the intended system behaviour.***

A system may fail in different forms, so the following concept of failure mode is introduced:

> ***Failure modes: The forms that the deviations from the system service may assume; failure modes are ranked according to failure severities (e.g. minor vs. catastrophic failures).***

At some point in time a system may fail, i.e., shows behaviour that deviates from the intended behaviour, and may return to the intended behaviour at some later point in time:

> ***System outage: The interval during which the system has failed, i.e. where the behaviour of the system deviated from its intended one.***
>
> ***System restoration: The transition from system failure to intended system behaviour.***
>
> ***Error: Part of the system state that deviated from the intended system state and could lead to system failure.***

It is important to note that many errors do not reach the system's external interfaces, hence do not necessarily cause system failure.

> ***Fault: The adjudged or hypothesized cause of an error; a fault is active when it causes an error, otherwise it is dormant.***

The prior presence of vulnerability, i.e., a fault that enables the existence of an error to possibly influence the system behaviour, is necessary such that a system fails.

The creation and manifestation mechanisms of faults, errors, and failures, depicted in Figure 1, is called "chain of threats". The chain of threats summarizes the causality relationship between faults, errors and failures. A fault activates (fault activation) in component A and generates an error; this error is successively transformed into other errors (error propagation) within the component

(internal propagation) because of the computation process. When some error reaches the service interface of component A, it generates a failure, so that the service delivered by A to component B becomes incorrect. The ensuing service failure of component A appears as an external fault to component B.

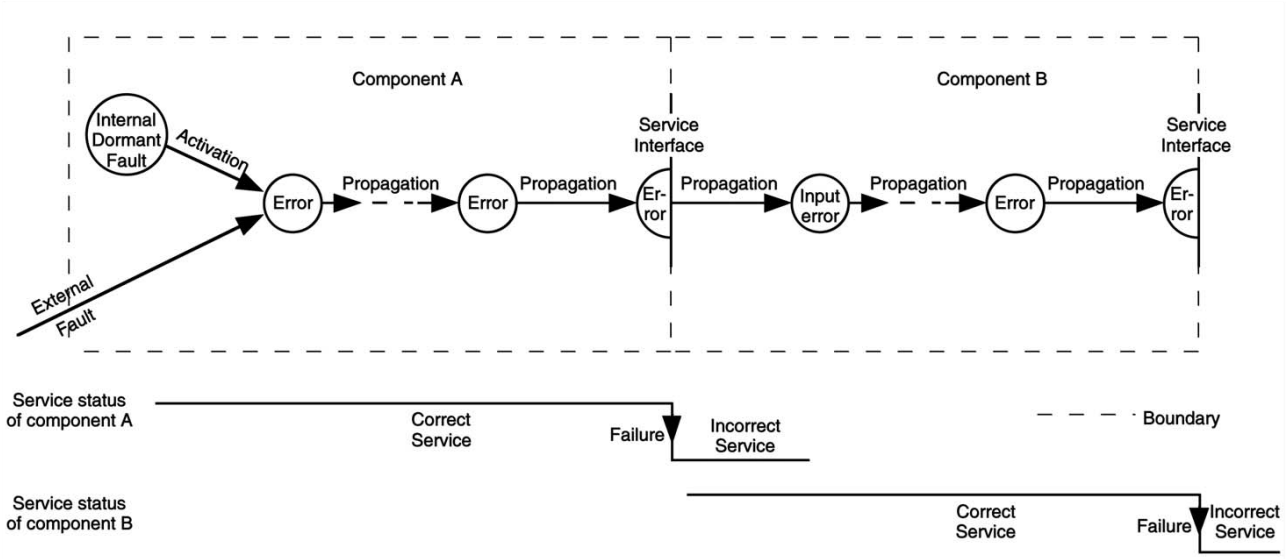

Figure 1 The chain of threats: a fault in component A activates and generates an error; errors propagate until component A fails; the failure of A appears as an external fault to B.[4]

9.2 DEPENDABILITY, ATTRIBUTES, AND ATTAINING DEPENDABILITY

> ***Dependability (original definition): The ability to deliver service that can justifiably be trusted.***

The above definition stresses the need for justification of "trust", so an alternate definition is given:

> ***Dependability (new definition): The ability to avoid failures that are more frequent and more severe than is acceptable.***

This last definition has a twofold role, because in addition to the definition itself it also provides the criterion for deciding whether the system is dependable or not.

Dependability is an integrating concept that encompasses the following dependability attributes:

> ***Availability: Readiness for service.***
>
> ***Reliability: Continuity of service.***
>
> ***Maintainability: The ability to undergo modifications and repairs.***
>
> ***Safety: The absence of catastrophic consequences on the user(s) and on the environment.***
>
> ***Integrity: The absence of improper system state alterations.***

A specialized secondary attribute of dependability is robustness.

> ***Robustness: Dependability with respect to external faults.***

The means to attain dependability (and security) are grouped into four major dependability categories:

> ***Fault prevention: The means to prevent the occurrence or introduction of faults.***
>
> ***Fault tolerance: The means to avoid service failures in the presence of faults.***

Fault removal: The means to reduce the number and severity of faults.

Fault forecasting: The means to estimate the present number, the future incidence, and the likely consequences of faults.

Fault prevention is part of general engineering and aims to prevent the introduction of faults during the development phase of the system, e.g. improving the development processes.

Fault tolerance aims to avoid the occurrence of failures by performing error detection (identification of the presence of errors) and system recovery (it transform a system state containing one or more errors into a state without detected errors and without faults that can be activated again) over time.

Fault removal can be performed both during the development phase, by performing verification, diagnosis and correction, and during the operational life, by performing corrective and preventive maintenance actions.

Fault forecasting is conducted by performing an evaluation of system behaviour with respect to fault occurrence of activation, using either qualitative evaluations (identifying, classifying and ranking the failure modes) or quantitative ones (evaluating in terms of probabilities the extent to which some of the attributes are satisfied).

The relationship among the above mentioned means are the following: fault prevention and fault tolerance aim to provide the ability to deliver a service that can be trusted, while fault removal and fault forecasting aim to reach confidence in that ability by justifying that the functional and the dependability and security specifications are adequate and that the system is likely to meet them.

9.3 SECURITY

Continuing with the concepts defined by Laprie [4], when addressing security an additional attribute needs to be considered: confidentiality.

Confidentiality: The absence of unauthorized disclosure of information.

Based on the above definitions, security is defined as follows:

Security: The composition of confidentiality, integrity, and availability; security requires in effect the concurrent existence of availability for authorized actions only, confidentiality, and integrity (with "improper" meaning "unauthorized").

We also consider the security definitions proposed by the CNSSI 4009 [1], which discusses security in terms of risk management. In this glossary, security is a condition that results from the establishment and maintenance of protective measures that enable an enterprise to perform its mission or critical functions despite risks posed by threats to its use of information systems.

Threat: Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, or other organizations through a system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.

A threat can be summarised as a failure, error or fault.

Vulnerability: Weakness in a system, system security procedures, internal controls, or implementation that could be exploited by a threat.

Vulnerabilities can be summarised as internal faults that enable an external activation to harm the system [4].

Risk is defined in terms of the impact and likelihood of a particular threat.

Risk: A measure of the extent to which an organization is threatened by a potential circumstance or event, and typically a function of 1) the adverse impacts that would arise if the circumstance or event occurs; and 2) the likelihood of occurrence.

Encryption using cryptography can be used to ensure confidentiality. The following definitions from Bruce Schneier's "Applied Cryptography" [8], have been modified to incorporate the definitions of *data* and *information* given in Section 4.1.

Encryption: The process of disguising data in such a way as to hide the information it contains.

Cryptography: The art and science of keeping data secure.

Data that has not been encrypted is referred to as plaintext or cleartext.

Plaintext: Unencrypted data.

Data that has been encrypted is called ciphertext.

Ciphertext: Data in its encrypted form.

The opposite of encryption is referred to as decryption.

Decryption: The process of turning ciphertext back into plaintext.

Encryption systems generally fall into two categories, asymmetric and symmetric, that are differentiated by the types of keys they use.

Key: A numerical value used to control cryptographic operations, such as decryption and encryption.

Symmetric Cryptography: Cryptography using the same key for both encryption and decryption.

Asymmetric cryptography is also known as public key cryptography.

Public Key Cryptography: Cryptography that uses a public-private key pair for encryption and decryption.

Symmetric cryptography uses the same key for encryption and decryption, called the symmetric key.

Symmetric Key: A cryptographic key that is used to perform both encryption and decryption.

Public key cryptography uses two kinds of keys, a public key and a private key.

Private Key: In an asymmetric cryptography scheme, the private or secret key of a key pair which must be kept confidential and is used to decrypt messages encrypted with the public key.

Generally, the private key is used for decryption and the public key is used for encryption. Depending on the cryptographic scheme, the public key can be used for authentication.

Public Key: A cryptographic key that may be widely published and is used to enable the operation of an asymmetric cryptography scheme. This key is mathematically linked with a corresponding private key.

Within cryptographic operations, Kerckhoff's principle states that only the cryptographic key (symmetric key for symmetric cryptography and the private key for public key cryptography) must remain secret in order for the system to be secure. This entails that the cryptographic algorithm and other settings can be public knowledge without compromising the secrecy of the encrypted information.

Controlling access to a system entails both authentication (verifying who is using the system) and authorisation (verifying what they can do) for subjects and objects.

> ***Subject: An active user, process, or device that causes information to flow among objects or changes the system state.***
>
> ***Object: Passive system-related devices, files, records, tables, processes, programs, or domain containing or receiving information. Access to an object implies access to the information it contains.***
>
> ***Authentication: The process of verifying the identity or other attributes claimed by or assumed of a subject, or to verify the source and integrity of data.***
>
> ***Authorisation: Authorisation is the mechanism of applying access rights to a subject. Authorising a subject is typically processed by granting access rights to them within the access control policy.***

The terms “authorisation” and “access control” are often used interchangeably. ***[6]***

> ***Access Control: Access control is concerned with providing control over security critical actions that take place in a system. Providing control over actions consists of explicitly determining either the actions that are permitted by the system, or explicitly determining the actions that are not permitted by the system.***

Authorisation is typically managed using an access control model.

> ***Access Control Model: An access control model captures the set of allowed actions as a policy within a system.***

Controlling what a subject can do within the system is normally managed by assigning permissions to that subject.

> ***Permission: Attributes that specify the access that subjects have to objects in the system.***

The security policy states what permissions are held by the users [10].

> ***Security Policy: Given identified subjects and objects, there must be a set of rules that are used by the system to determine whether a given subject can be permitted to gain access to a specific object. This is called the security policy.***

In an access control system, the security policy is enforced by what is called the reference monitor.

> ***Reference Monitor: A reference monitor represents the mechanism that implements the access control model. A reference monitor is defined as: An access control concept that refers to an abstract machine that mediates all accesses to objects by subjects.***

The concept of trust is defined in terms of failures and the ability to break the security policy ***[7]***.

> ***Trusted System: A trusted system or component is one whose failure can break the security policy.***
>
> ***Trusted Computing Base: The set of components (hardware, software, human,…) whose correct functioning is sufficient to ensure that the security policy is enforced, or, more vividly, whose failure could cause a breach of the security policy.***

Anderson also distinguishes between *trusted* and *trustworthy* [7]. The failure of a trusted system could break the security policy, whereas a trustworthy system will not fail (and thus break the security policy).

> ***Trustworthy System: A system or component that warrants trust because the system or component's behaviour can be validated in some convincing way, such as through formal analysis or code review.***

In the next phase of the AMADEOS project, we will investigate how to build a secure and dependable SoS architecture and develop best practices for such secure and dependable systems.

10 EVOLUTION AND DYNAMICITY

10.1 BASIC CONCEPTS

Large scale Systems-of-Systems (SoSs) tend to be designed for a long period of usage (10 years+). Over time, the demands and the constraints put on the system will usually change, as will the environment in which the system is to operate. The AMADEOS project studies the design of systems of systems that are not just robust to dynamicity (short term change), but to long term changes as well. This Section addresses a number of terms related to the evolution of SoSs.

> ***Evolution: Process of gradual and progressive change or development, resulting from changes in its environment (primary) or in itself (secondary).***

Although the term evolution in other contexts does not have a positive or negative direction, in the SoSs context, evolution refers to maintaining and optimizing the system - a positive direction, therefore.

> ***Managed evolution: Evolution that is guided and supported to achieve a certain goal.***

For SoSs, evolution is needed to cope with changes. Managed evolution refers to the evolution guidance. The goal can be anything like performance, efficiency, etc. The following two definitions further detail managed evolution for SoSs:

> ***Managed SoS evolution: Process of modifying the SoS to keep it relevant in face of an ever-changing environment.***

This is Primary evolution; examples of environmental changes include new available technology, new business cases / strategies, new business processes, changing user needs, new legal requirements, compliance rules and safety regulations, changing political issues, new standards, etc.

> ***Unmanaged SoS evolution: Ongoing modification of the SoS that occurs as a result of ongoing changes in (some of) its CSs.***

This is Secondary evolution; examples of such internal changes include changing circumstances, ongoing optimization, etc. This type of evolution may lead to unintended emergent behaviour, e.g., due to some kind of “mismatch” between Constituent Systems (CSs) (see Section 2.3).

Managed SoS evolution is due to changes in the environment (see Section 2.2).

The goal of managed evolution in AMADEOS is maximizing business value:

> ***Business value: Overarching concept to denote the performance, impact, usefulness, etc. of the functioning of the SoS.***

Quantizing business value is difficult since it is a multi-criteria optimization problem. Aiming for Pareto optimality, various aspects (measured by utility functions) are weighted on a case-by-case basis.

System performance is a key term in the concept of business value:

> ***System performance: The combination of system effectiveness and system efficiency.***
>
> ***System effectiveness: The system's behaviour as compared to the desired behaviour.***
>
> ***System efficiency: The amount of resources the system needs to act in its environment.***

For the last definition, it is important to understand what system resources are in the area of SoS:

> ***System resources: Renewable or consumable goods used to achieve a certain goal. E.g., a CPU, CPU-time, electricity.***

Dynamicity: The property of an entity that is constantly changing in terms of offered services, built-in structure and interactions with other entities.

Linked to dynamicity and to the control strategy shifting from a central to an autonomous paradigm, is the concept of *reconfigurability*.

Reconfigurability: The ability to repeatedly change and rearrange the components of an entity in a cost-effective way.

10.2 SCENARIO-BASED REASONING

The management of any activity entails a number of processes that are continuously executed. These processes can be summarised with reference to the Observe, Orient, Decide and Act (OODA) loop[2]. In AMADEOS these steps are applied to the *managed evolution* of an SoS. Given the context of an SoS, the following assumptions can be reasonably made:

1. It is impossible to observe the *entire* state of the SoS. Therefore, observations about the SoS are partial (i.e., incomplete), and possibly uncertain and/or conflicting. This may lead to a number of possible interpretations or mental perspectives on the (current) state of the SoS and its possible developments.
2. It is impossible to *precisely* determine the *best* course of actions to take, given a certain interpretation or mental perspective on the state of the SoS, and to predict the effects of management actions. This may also lead to a number of possible outcomes of the decisions made and actions taken.

Given the predominant uncertainties in the processes above, a structured technique that can assist in the Orient and Decide phases of the managed SoS evolution and address those uncertainties is introduced, namely Scenario-Based Reasoning (SBR).

Scenario-Based Reasoning (SBR): Systematic approach to generate, evaluate and manage different scenarios in a given context.

Scenarios, in turn, are “what-if” stories that give alternative accounts of a situation and may include assumptions as well as real pieces of information.

Scenario: A scenario is a projected or imagined sequence of events describing what could possibly happen in the future (or have happened in the past).

As such, scenarios should be plausible (not go beyond what is possible), consistent (with no contradictions between parts) and coherent (with explicit logical connections between events).

Scenarios can be used to deal with uncertainties regarding a situation, supporting processes of situation assessment and decision making.

Situation assessment: Situation assessment is the process of achieving, acquiring or maintaining situation awareness.

Decision making: Decision making is the process resulting in the selection of a course of action among several alternative possibilities.

Situation awareness, on the other hand, is a state of knowledge on the situation.

Situation awareness: The perception of the context in a given situation, the comprehension of their meaning and the projection of their status in the near future.

[2] Attributed to USAF Colonel John Boyd, who gave numerous presentations on this topic. No original papers written by him. For more information, see http://en.wikipedia.org/wiki/OODA_loop

In order to generate scenarios representing e.g. multiple states of SoS situation awareness or the possible effects of management actions on the SoS, SBR makes use of observations of some relevant states and domain models in combination with SoS inference processes.

> ***Domain models: Models that represent relations between different items of knowledge in a given domain.***
>
> ***SoS inference processes: Processes that map observations about the SoS to estimates about states that are relevant for decision making or planning but cannot be directly observed.***

The domain models represent the general knowledge of the domain, i.e. knowledge that is independent of the specific system that is reasoned about. An example of a domain model is a causal model.

> ***Causal model: Abstract model describing the causal dependencies between relevant variables in a given domain.***

The causal model must express more than correlation relations, since correlation does not imply causation. The causal dependencies in causal models can be made explicit through the use of causal graphs.

> ***Causal graphs: Directed graphs representing the cause-effect relations between the variables (nodes) in the graph.***

In summary, the domain models represent the scenario structure whereas the (SoS) inference processes are used to derive values for the variables in the scenario.

The management of generated set scenarios for a given situation involves the processes of scenario pruning and scenario updating.

> ***Scenario pruning: Scenario pruning is the process of discarding scenarios that include incorrect or unlikely information.***
>
> ***Scenario updating: Scenario updating addresses the problem of how to deal with newly available information, not present in the generated set of scenarios.***

In order to support the managed SoS evolution, modifications to the SoS may be proposed in the form of decision alternatives.

> ***Decision alternative: Alternative course-of-action that may be chosen in the process of decision making.***

For the managed-evolution of SoSs each decision alternative is (likely to be) a plan of actions, such as (but not restricted to): adjustments to monitoring, configuration changes of the SoS or its CSs, functioning changes, behaviour changes, add/remove/update CSs, change environment, etc.

The evaluation of each decision alternative is typically performed taking into account the decision problem's overall goal (e.g. maximizing business value of the SoS). Typically, the overall goal is broken down into (multiple) criteria, resulting in a multi-criteria optimization problem that may be addressed with Multi-Criteria Decision Analysis (MCDA).

> ***Multi-Criteria Decision Analysis (MCDA): MCDA is a sub-discipline of operations research that explicitly considers multiple criteria in decision-making, allowing the evaluation of one or more decision alternatives in light of the multiple criteria.***

With SBR it is possible, in a structured and traceable manner, to reason about uncertain and missing information, as well as to explore possible different futures, including futures in which given decision alternatives are effectuated. The MCDA approach allows the evaluation of these decision alternatives for a number of different scenarios.

11 SYSTEM DESIGN AND TOOLS

SoSs can have a very complex architecture. They are constituted by several CSs which interact with each other. Due to their complexity, well defined design methodologies should be used, in order to avoid that some SoS requirement is not fulfilled and to ease the maintainability of the SoS.

11.1 ARCHITECTURE

We start defining what is it an architecture.

> ***Architecture: The manner in which the components of a system are organized and integrated. It defines the structure, behaviour, and more views of a system.***

The architecture represents a more static view about how the components constitute the system. The architecture of a system can have some variants or even can vary during its operation.

Then this adaptability of the architecture is translated into three more concepts.

> ***Evolvable architecture: An architecture that is adaptable and then is able to incorporate known and unknown changes in the environment or in itself.***
>
> ***Flexible architecture: Architecture that can be easily adapted to a variety of future possible developments.***
>
> ***Robust architecture: Architecture that performs sufficiently well under a variety of possible future developments.***

The architecture then involves several components which interact with each other. The place where they interact is defined as interface (see Section 2.2).

During the development lifecycle of a system, we start from conceptual thoughts which are then translated into requirements, which are then mapped into an architecture. The process that brings designers to define a particular architecture of the system is called design.

> ***Design: The process of defining an architecture, components, modules and interfaces of a system to satisfy specified requirement.***

In the AMADEOS context, design is a verb, architecture is a noun. The people who perform the design are designers.

> ***Designer: An entity that specifies the structural properties of a design object.***

There are several methodologies to design a system.

> ***Hierarchical Design: An approach to design that leverages the natural logical hierarchy.***

This methodology is useful to decrease the degree of the complexity of a system and to ease its maintainability. In particular, in the hierarchical design the system can be split in different subsystems that can be grouped in modules.

> ***Module: A set of standardized parts or independent units that can be used to construct a more complex structure.***

The modularity is the technique that combines these modules in order to build a more complex system.

> ***Modularity: Engineering technique that builds larger systems by combining smaller subsystems.***

In the context that an SoS shall deal with evolving environments, then also design methodologies focused on evolution shall be defined.

> ***Design for evolution: Exploration of forward compatible system architectures, i.e. designing applications that can evolve with an ever-changing environment, e.g. Internet applications that are forward compatible given changes in business processes and strategies as well as in technology (digital, genetic, information-based and wireless). Principles of evolvability include modularity, updateability and extensibility. Design for evolution aims to achieve robust and/or flexible architectures.***

In the context of SoS, design for evolution can be reformulated in the following manner.

> ***Design for evolution in the context of SoS: Design for evolution means that we understand the user environment and design a large SoS in such a way that expected changes can be accommodated without any global impact on the architecture. 'Expected' refers to the fact that changes will happen, it does not mean that these changes themselves are foreseeable.***

In addition during the system development lifecycle some activities to verify if the architecture developed during the design process is compliant and if it fulfills the requirements of the system are foreseen. Verification of design is an important activity especially in critical systems and several methodologies are defined to perform it. In particular there is a methodology which has in mind verification since the design phase.

> ***Design for testability: The architectural and design decisions in order to enable to easily and effectively test our system.***

Then we have two methodologies to perform design verification:

> ***Design inspection: Examination of the design and determination of its conformity with specific requirements.***
>
> ***Design walkthrough: Quality practice where the design is validated through peer review.***

In order to perform all these activities some useful tools can be used to help the designers and verifiers job.

Starting from the item definition we can now define the concept of a tool:

> ***Tool: A tool is any item that can be used to achieve a goal.***

12 GOVERNANCE

The underlying purpose of SoS governance is to ensure that the interoperation among constituent systems will achieve SoS goals.

> ***Governance: Theoretical concept referring to the actions and processes by which stable practices and organizations arise and persist. These actions and processes may operate in formal and informal organizations of any size; and they may function for any purpose.***

In the context of SoS, governance can be implemented in a collaborative fashion, due to the independence of the constituent systems.

> ***Collaborative governance: Process and a form of governance in which participants (parties, agencies, stakeholders) representing different interests are collectively empowered to make a policy decision or make recommendations to a final decision-maker who will not substantially change consensus recommendations from the group.***

The governance actions and processes are initiated by an administrative domain part of the organisation.

> ***Administrative domain: An organisation that controls resources and constituent systems that are part of the SoS (or: going to be part of the SoS).***

Administrative domain has authority to make changes to configuration of resources & systems, and their connections to other resources & systems in other administrative domains.

> ***Authority: The relationship in which one party has the right to demand changes in the behaviour or configuration of another party, which is obliged to conform to these demands.***

13 QUALITY METRICS AND ATTRIBUTES

Including quality metrics and attributes in the conceptual model for the design of System-of-Systems (SoS) is challenging. In fact, on the one hand, SoS are systems themselves, and so dependability, resilience, security levels, quality of service and performance attributes and metrics normally defined for a Constituent System (CS) can be in principle used for SoS functional correctness verification, as well. On the other hand, SoSs have peculiarities that call for specific quality attributes and metrics, such as the interaction of autonomous and independently evolving CSs and the previously defined emergence.

> ***Quality: The standard of something as measured against other things; the degree of excellence of something.***
>
> ***Quality of Service: The ability of a system to meet certain requirements for different aspects of the system like performance, dependability, evolvability, security or cost; possibly expressed in terms of levels and quantitatively evaluated through metrics.***
>
> ***Metric: Property or indicator used to quantitatively describe an aspect of the system, like throughput for performance or availability for dependability.***
>
> ***Resilience: Persistence of service delivery that can justifiably be trusted, when facing changes. Changes here may refer to unexpected failures, attacks or accidents (e.g., disasters). Changes may also refer to increased loads. Resilience thus deals with conditions that are outside the design envelope whereas other dependability metrics deal with conditions within the design envelope [3].***
>
> ***Security Level: Specification of the level of security to be achieved through the establishment and maintenance of protective measures.***

It is useful to look at Table I, in order to single out a set of attributes that are specific for SoS and their peculiarities compared to old-monolithic systems.

As regards *testing*, the need for continuous testing vs. established test phases, and the frequency and nature of faults becoming *normal* rather than *exceptional* events, call for *testability* and (the already defined) *robustness* of SoS. Also, *repeatability* is more difficult to be granted.

> ***Testability: Property of a system to support testing.***
>
> ***Repeatability: Condition of measurement, out of a set of conditions that includes the same measurement procedure, same operators, same measuring system, same operating conditions and same location, and replicate measurements on the same or similar objects over a short period of time [5].***

Similarly, the fact that *requirements* and *specifications* are not fixed anymore, but subject to frequent changes poses dynamicity (cf. Section 10.1) as a relevant attribute.

> ***Requirement: A statement that identifies a necessary attribute, capability, characteristic, or quality of a system.***

According to the definition of SoS: "*An SoS is an integration of a finite number of constituent systems which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal,"* the role of communication is central, as well as the notion of time. Consequently, information *integrity* and *consistency*, and communication *timeliness* are vital attributes of an SoS.

> ***Consistency: The property of a set of entities that see the same data at the same time.***

Due to the heterogeneity of SoS natures and scopes, it is difficult to generalize attributes and metrics, as relevant attributes of different SoSfs may differ from one another. So, we will specify

quality attributes and metrics presented in this Section in the conceptual model and add domain-specific attributes and metrics, on a use-case basis (see WP4).

Contract: Agreement between two or more parties, where one is the customer and the others are service providers. This can be a legally binding formal or an informal "contract". It can be expressed in terms of objectives.

Objective: Values for the quality metrics to be attained.

Quality of Experience/ Quality of Perception: A subjective measure of a user's experience/perception with a system and satisfaction for a system's quality.

14 REFERENCES

[1] *CNSS Instruction No. 4009: National Information Assurance (IA) Glossary*, April 26, 2010. Retrieved from Committee on National Security Systems: http://www.ncix.gov/publications/policy/docs/CNSSI_4009.pdf.

[2] Kopetz, H. "*Real-Time Systems: Design Principles for Distributed Embedded Applications", 2nd ed., Springer 2011.*

[3] Laprie J., LAAS-CNRS, "Resilience for the scalability of dependability", Proc. ISNCA 2005, pp. 5-6.

[4] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE Trans. on Dependable and Secure Computing, vol. 1, no. 1, Jan. –Mar. 2004.

[5] BIPM, Joint Committee for Guides in Metrology (JCGM). International Vocabulary of Metrology. Third Edition.

[6] T.B. Quillinan, "Secure Naming for Distributed Computing using the Condensed Graph Model", University of Ireland, 2006.

[7] Security Engineering, Ross Anderson. Second Edition, Wiley. 2008

[8] Schneier, B. "Applied Cryptography". Second edition, Wiley. 1996.

[9] J. Rushby, "The Design and Verification of Secure Systems" In the 8th ACM Symposium on Operating System Principles (SOSP), pp 12—21. Asilomar, CA, December 1981.

[10] The Department of Defence Trusted Computer System Evaluation Criteria (TCSEC) (commonly known as the Orange Book).

[11] "Trust" Oxford English Dictionary, Retreived on 28.04.2014 from: http://www.oxforddictionaries.com/definition/english/trust?q=Trust

[12] Jackson, D et al. *"Software for Dependable Systems: Sufficient Evidence?"*, National Academic Press, Washington, 2007.

[13] Boulding, K. *"The Image: Knowledge in Life and Society",* Univ. of Michigan Press. 2010.

[14] "Dependability" Oxford English Dictionary, Retrieved on 02.05.2014 from: http://www.oxforddictionaries.com/definition/english/dependable

[15] Vigotsky, L.S. "*Thought and Language"*. MIT Press. Boston, 1962.

[16] Hayakawa, S.I. "*Language in Thought and Action"*. Mariner Books, 1991.

[17] Wikipedia, "*Conceptual Model in Computer Science"*. 2014.

[18] Chakravarty, A. "*Scientific Realism".* Stanford Encyclopedia of Philosophy, 2011.

[19] Popper, K. Thr*ee Worlds. "The Tanner Lecture on Human Values".* Univ. of Michigan, 1978.

[20] *"Final Version of the DSOS Conceptual Model*". Technical Report CS-TR-782, University of Newcastle upon Tyne. GB, 2003.

[21] *"D1.1 – SoSs, Commonalities and Requirements".* AMADEOS Project. 2014.

[22] Simon. H. "*The Science of The Artificial".* MIT Press, 1969.

[23] Jamshidi, M. "*Systems of Systems Engineering—Innovations for the 21st century".* Wiley and Sons, 2009.

[24] Dahman, J.S. and Baldwin, K.J. *"Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering*". Proc. of 2nd Annual IEEE Systems Conference. Montreal. IEEE Press. 2008.

[25] Withrow, G.J. "*The Natural Philosophy of Time"*. Oxford Science Publications. 1990.

[26] Winfree, A.T. "*The Geometry of Biological Time"*. Springer Verlag. 2001.

[27] Decotignie, J.D. "*Which Network for which Application?"* in: *The Industrial Communication Technology Handbook.* Ed.: R. Zuwarski, Taylor and Francis, Boca Raton, US, 2005.

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

[29] Lombardi, M.A. "*The Use of GPS Disciplines Oscillators as Primary Frequency Standards for Calibration and Metrology Laboratories"*. Measure—The Journal of Measurement Science. pp. 56-65. 2008.

[30] US Government Accountability Office: "*GPS Disruptions: Efforts to Assess Risk to Critical Infrastructure and Coordinate Agency Actions Should be Enhanced".* Washington, GAO -14-15. 2013.
[31] Zins, C. "*Conceptual Approaches for Defining Data, Information and Knowledge"*. Journal of the American Society for Information Science and Technology. Vol 58 (4). pp. 479-493. 2007.
[32] Kopetz, H. *A. "Conceptual Model for the Information Transfer in Systems of Systems".* Proc. of ISORC 2014. Reno, Nevada. IEEE Press. 2014.
[33] Floridi, L. "*Is Semantic Information Meaningful Data?"* Philosophy and Phenomenological Research. Vol 60. No. 2.Pp.351-370. 2005.
[34] Glanzberg, M. "*Truth"*. Stanford Encyclopedia on Philosophy. 2013.
[35] World Wide Web Consortium. Extensible Markup Language. URL: http://www.w3.org/XML/ Retrieved on April 18, 2013.
[36] Aviation Safety Network. Accident Description. URL: http://aviationsafety. net/database/record.php?id=19920120-0 retrieved on June 10, 2013.
[37] Siegel, J. "*CORBA 3 Fundamentals and Programming"*. John Wiley, 2000.
[38] Stephan, A., *"Emergence—A Systematic View on its Historical Facets".* In: Beckerman, E at al. Editors. Emergence or Reduction? Walter de Gruyter, Berlin, 1992.
[39] Beckerman, A et al. (ed.) *"Emergence or Reduction—Essays on the Progress of Non-reductive Physicalism".* Walter de Gruyter, Berlin,1992.
[40] Clayton, P., and Davies, P. "*The Reemergence of Emergence".* Oxford University Press, 2006.
[41] Kim, J. *"Emergence: Core Ideas and Issues".* Retrieved from: http://cs.calstatela.edu/~wiki/images/b/b1/Emergence-Core_ideas_and_issues.pdf. Published online on August 9, 2006.
[42] Bedau, M.A. and Humphreys, P., *"Emergence, Contemporary Readings in Philosophy and Science"*. MIT Press, 1968.
[43] Koestler, A. *"The Ghost in the Machine".* Hutchinson. London. 1976.
[44] O'Connor, T. *"Emergent Properties".* Stanford Encyclopedia of Philosophy. 2012.
[45] Davies, C.W. *"The Physics of Downward Causation"*. The Reemergence of Emergence. Ed. By Clayton P and Davies, P. Oxford University Press. 2006.
[46] McLaughlin, B and Bennet, K. "*Supervenience"*. Stanford Encyclopedia of Philosophy. 2011.
[47] Stanislaw H. Zak. *"Systems and control"*. Oxford University Press. 2003.

15 GLOSSARY

Concept	Definition
Absolute Timestamp	An absolute timestamp of an event is the timestamp of this event that is generated by the reference clock.
Acceptance Test	A test that determines if a state in the problem space is a member of the solution set.
Access Control	Access control is concerned with providing control over security critical actions that take place in a system. Providing control over actions consists of explicitly determining either the actions that are permitted by the system, or explicitly determining the actions that are not permitted by the system.
Access Control Model	An access control model captures the set of allowed actions as a policy within a system.
Accuracy	The accuracy of a clock denotes the maximum offset of a given clock from the external time reference during the IoD, measured by the reference clock.
Acknowledged SoS	Independent ownership of the CSs, but cooperative agreements among the owners to an aligned purpose.
Action	The execution of a program by a computer or a protocol by a communication system.
Action Sequence	A sequence of actions, where the end-signal of a preceding action acts as the start signal of a following action.
Activity Interval	The interval between the start signal and the end signal of an action or a sequence of related actions.
Administrative domain	An organisation that controls resources and constituent systems that are part of the SoS (or: going to be part of the SoS).
Architectural Style	The set of explicit or implicit rules and conventions that determine the structure and representation of the internals of a system, its data and protocols.
Architecture	The manner in which the components of a system are organized and integrated. It defines the structure, behaviour, and more views of a system.
Arrival Instant	The instant when the first bit of a message arrives at the receiver.
Artifact	An entity that has been intentionally produced by a human for a certain purpose.
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.
Attribute	A refinement of a property.
Authentication	The process of verifying the identity or other attributes claimed by or assumed of a subject, or to verify the source and integrity of data.
Authorisation	Authorisation is the mechanism of applying access rights to a subject. Authorising a subject is typically processed by granting access rights to them within the access control policy.
Authority	The relationship in which one party has the right to demand changes in the behaviour or configuration of another party, which is obliged to conform to

	these demands.
Autonomous System	A system that can provide its services without guidance by another system.
Availability	Readiness for service.
Back-pressure flow control	A message control schema, where the receiver exerts back pressure on the sender to ensure that the speed of the sender will not outpace the speed of the receiver and thus lead to queue overflow.
Behaviour	The timed sequence of the effects of input and output actions that can be observed at an interface of a system.
Business value	Overarching concept to denote the performance, impact, usefulness, etc. of the functioning of the SoS.
Causal graphs	Directed graphs representing the cause-effect relations between the variables (nodes) in the graph.
Causal model	Abstract model describing the causal dependencies between relevant variables in a given domain.
Causal order	A causal order among a set of events is an order that reflects the cause-effect relationships among the events.
Ciphertext	Data in its encrypted form.
Clock	A (digital) clock is an autonomous system that consists of an oscillator and a register. Whenever the oscillator completes a period, an event is generated that increments the register.
Closed System	A system that is not interacting with its environment during a given IoD.
Collaborative governance	Process and a form of governance in which participants (parties, agencies, stakeholders) representing different interests are collectively empowered to make a policy decision or make recommendations to a final decision-maker who will not substantially change consensus recommendations from the group.
Collaborative SoS	Voluntary interactions of independent CSs to achieve a goal that is beneficial to the individual CS.
Communication Action	An action that is characterized by the execution of a communication protocol by a communication system.
Communication Protocol	The set of rules that govern a communication action.
Component	A subsystem of a system, the internal structure of which is of no interest.
Computational Action	An action that is characterized by the execution of a program by a machine.
Concept	A category that is augmented by a set of beliefs about its relations to other categories, i.e., existing knowledge, is called a concept.
Concise State	The state of a system is considered concise if the size of the declared ground state is at most in the same order of magnitude as the size of the system's largest input message.
Confidentiality	The absence of unauthorized disclosure of information.
Consistency	The property of a set of entities that see the same data at the same time.
Constituent System	An autonomous subsystem of an SoS, consisting of computer systems and

(CS)	possibly of controlled objects and/or human role players that that interact to provide a given service.
Constraint	A restriction in the problem space.
Construct	A non-physical entity, a product of the human mind.
Consume/Produce (CP) paradigm	At the sender, the communication system consumes the message from a sender queue and at the receiver the communication system adds the received message to a receiver queue.
Context	The set of cultural circumstances, conventions or facts, and the time that surround and have a possible influence on a particular thing, construct, event, situation, system, etc. in the UoD.
Contract	Agreement between two or more parties, where one is the customer and the others are service providers. This can be a legally binding formal or an informal "contract". It can be expressed in terms of objectives.
Control flow	The flow of control signals when executing a protocol.
Cryptography	The art and science of keeping data secure.
Cyber-Physical System (CPS)	A system consisting of a computer system (the cyber system), a controlled object (a physical system) and possibly of interacting humans.
Cycle	A temporal sequence of significant events that, upon completion, arrives at a final state that is related to the initial state, from which the temporal sequence of significant events can be started again.
Data	A data item is an artifact, a pattern, created for a specified purpose.
Data flow	The flow of the payload data of a message from a sender to the receivers.
Data Frame	A valued data structure produced by a periodic system.
Data stream	A sequence of data frames produced by a periodic system.
Datagram	A best effort message transport service for the transmission of sporadic messages from a sender to one or many receivers.
Deadline	An instant when a computational action or a communication action must be completed.
Decision alternative	Alternative course-of-action that may be chosen in the process of decision making.
Decision making	Decision making is the process resulting in the selection of a course of action among several alternative possibilities.
Decision Problem	A problem to select an alternative out of a set of given alternatives.
Declared Ground State	A declared data structure that contains the relevant ground state of a given application at the ground state instant.
Decryption	The process of turning ciphertext back into plaintext.
Dependability	This is the ability to avoid failures that are more frequent and more severe than is acceptable.
Design	The process of defining an architecture, components, modules and interfaces of a system to satisfy specified requirement.
Design for evolution	Exploration of forward compatible system architectures, i.e. designing applications that can evolve with an ever-changing environment, e.g. Internet applications that are forward compatible given changes in business processes and strategies as well as in technology (digital, genetic,

	information-based and wireless). Principles of evolvability include modularity, updateability and extensibility. Design for evolution aims to achieve robust and/or flexible architectures.
Design for evolution in the context of SoS	Design for evolution means that we understand the user environment and design a large SoS in such a way that expected changes can be accommodated without any global impact on the architecture. 'Expected' refers to the fact that changes will happen, it does not mean that these changes themselves are foreseeable.
Design for testability	The architectural and design decisions in order to enable to easily and effectively test our system.
Design inspection	Examination of the design and determination of its conformity with specific requirements.
Design walkthrough	Quality practice where the design is validated through peer review.
Designer	An entity that specifies the structural properties of a design object.
Deterministic Behaviour	A system behaves deterministically if, given an initial state at a defined instant and a set of future timed inputs, the future states, the values and instants of all future outputs are entailed.
Diagnosis Problem	A problem to find the cause of observed symptoms.
Directed SoS	An SoS with a central managed purpose and central ownership of all CSs.
Domain models	Models that represent relations between different items of knowledge in a given domain.
Downward Causation	The phenomenon that some novel higher-level properties cannot be reduced to mere physical phenomena and have causal powers to control the lower-level process from which they emerge.
Drift	The drift of a physical clock is a quality measure describing the frequency ratio between the physical clock and the reference clock.
Drift Rate	|Drift – 1|
Duration	The length of an interval.
Dynamic Attribute	An attribute of a property that can change its value during the IoD.
Dynamicity	The property of an entity that is constantly changing in terms of offered services, built-in structure and interactions with other entities.
Emergence	A phenomenon of a whole at the macro-level is emergent if and only if it is new with respect to the non-relational phenomena of any of its proper parts at the micro level.
Encrypted data (called cipher-text)	Data that has been subjected to a transformation (encryption) such that an unauthorized user cannot easily interpret the encrypted data.
Encryption	The process of disguising data in such a way as to hide the information it contains.
End signal	An event that is produced by the termination of an action.
Entity	Something that exists as a distinct and self-contained unit.
Entourage of a Cyber-Physical System (CPS)	The entourage is composed of those entities of a CPS (e.g., the role playing human, controlled object) that are external to the cyber system of the CPS but are considered an integral part of the CPS.

Environment of a System	The entities and their actions in the UoD that are not part of a system but have the capability to interact with the system.
Epoch	An instant on the time-line chosen as the origin for time-measurement.
Error	Part of the system state that deviated from the intended system state and could lead to system failure.
Event	A happening at an instant.
Event Variable	A variable that holds information about some change of state at an instant.
Event-triggered (ET) Action	An action where the start signal is derived from an event other than the progression of time.
Evolution	Process of gradual and progressive change or development, resulting from changes in its environment (primary) or in itself (secondary).
Evolutionary System	A system where the external system boundary is dynamic (i.e., changes during the IoD).
Evolvable architecture	An architecture that is adaptable and then is able to incorporate known and unknown changes in the environment or in itself.
Execution Time	The duration it takes to execute a specific action on a given computer.
Explanation	The explanation of the data establishes the links between data and already existing concepts in the mind of a human receiver or the rules for handling the data by a machine.
Explicit flow control	After having sent a message, the sender receives a control message from the receiver informing the sender that the receiver has processed the sent message.
External Clock Synchronization	The synchronization of a clock with an external time base such as GPS.
Failure	The actual system behaviour deviation from the intended system behaviour.
Failure modes	The forms that the deviations from the system service may assume; failure modes are ranked according to failure severities (e.g. minor vs. catastrophic failures).
Fault	The adjudged or hypothesized cause of an error; a fault is active when it causes an error, otherwise it is dormant.
Fault forecasting	The means to estimate the present number, the future incidence, and the likely consequences of faults.
Fault prevention	The means to prevent the occurrence or introduction of faults.
Fault removal	The means to reduce the number and severity of faults.
Fault tolerance	The means to avoid service failures in the presence of faults.
Firm Deadline	A deadline for a result is firm if the result has no utility after the deadline has passed.
Flexible architecture	Architecture that can be easily adapted to a variety of future possible developments.
Flow control	The control of the flow of messages from the sender to the receiver such that the sender does not outpace the receiver.

Formal Problem	A problem in a well-defined problem space.
Function	The intended behaviour of a stateless system.
Goal State	An element of the solution set.
Governance	Theoretical concept referring to the actions and processes by which stable practices and organizations arise and persist. These actions and processes may operate in formal and informal organizations of any size; and they may function for any purpose.
GPSDO (Global Positioning Disciplined Oscillator)	The GPSDO synchronizes its time signals with the information received from a GPS receiver.
Granularity/Granule of a clock	The duration between two successive ticks of a clock is called the granularity of the clock or a granule of time.
Ground State	At a given level of abstraction, the ground state of a cyclic system is a state at an instant when the size of the state space is at a minimum relative to the sizes of the state space at all other instants of the cycle.
Ground State Instant	The instant of the ground state in a cyclic system.
Hard Deadline	A deadline for a result is hard if a catastrophe can occur in case the deadline is missed.
Hierarchical Design	An approach to design that leverages the natural logical hierarchy.
Holarchy	A structure where holons at one level interact horizontally to form a novel holon at the next higher level.
Holdover	The duration during which the local clock can maintain the required precision of the time without any input from the GPS.
Holon	A two-faced entity in a non-formal hierarchy that externally (upwards and horizontally) acts as a whole but internally (downwards) is established by the interactions of its parts (i.e., the interactions among the holons of the level below).
Homogenous System	A system where all sub-systems adhere to the same architectural style.
Idempotent Action	An action is idempotent if the effect of executing it more than once has the same effect as of executing it only once.
Ill-structured Problem	A problem where either the initial state and/or the solution states, and/or the acceptance test and/or constraints, are not well defined.
Implicit flow control	The sender and receiver agree a priori on a maximum send rate. The sender commits to never send messages faster than the agreed send rate and the receiver commits to accept all messages that the sender has sent.
Information	A proposition about the state of or an action in the world.
Initial State	(i) an existing deficient state of affairs that needs a solution or (ii) a recognized opportunity that should be exploited or (iii) a formal statement of a question (academic story problem).
Input Action	An action that reads or consumes input data at an interface.
Input data	Data that is used as an input to a system.

Instant	A cut of the timeline.
Integrity	The absence of improper system state alterations.
Interface	A point of interaction of a system with another system or with the system environment.
Internal Clock Synchronization	The process of mutual synchronization of an ensemble of clocks in order to establish a global time with a bounded precision.
Interval	A section of the timeline between two instants.
Interval of Discourse (IoD)	The Interval of Discourse specifies the time interval that is of interest when dealing with the selected view of the world.
Intra-ordinal Law	A new law that deals with the emerging phenomena at the macro level.
Irrevocable Action	An action that cannot be undone.
Itom	An Itom (Information Atom) is a tuple consisting of data and the associated explanation of the data.
Jitter	The duration between the minimal transport duration and the maximum transport duration.
Key	A numerical value used to control cryptographic operations, such as decryption and encryption.
Legacy System	An existing operational system within an organization that provides an indispensable service to the organization.
Maintainability	The ability to undergo modifications and repairs.
Managed evolution	Evolution that is guided and supported to achieve a certain goal.
Managed SoS evolution	Process of modifying the SoS to keep it relevant in face of an ever-changing environment.
Message	A data structure that is formed for the purpose of the timely exchange of information among computer systems.
Message assertion	A message assertion is a predicate on the values of a message and relevant state variables that defines an application specific acceptance criterion.
Meta Data	Data that describes the meaning of object data.
Metric	Property or indicator used to quantitatively describe an aspect of the system, like throughput for performance or availability for dependability.
Modularity	Engineering technique that builds larger systems by combining smaller subsystems.
Module	A set of standardized parts or independent units that can be used to construct a more complex structure.
Monolithic System	A system is called monolithic if distinguishable services are not clearly separated in the implementation but are interwoven.
Multi-Criteria Decision Analysis (MCDA)	MCDA is a sub-discipline of operations research that explicitly considers multiple criteria in decision-making, allowing the evaluation of one or more decision alternatives in light of the multiple criteria.
Non-Sparse Events	Events that occur in the passive interval of the sparse time.
Object	Passive system-related devices, files, records, tables, processes, programs, or domain containing or receiving information. Access to an

	object implies access to the information it contains.
Object Data	Data that is the object of description by meta data.
Objective	Values for the quality metrics to be attained.
Observation of an Entity	An atomic structure consisting of the name of the entity, the name of the property, the value of the property and the timestamp denoting the instant of observation.
Offset of events	The offset of two events denotes the duration between two events and the position of the second event with respect to the first event on the timeline.
Open System	A system that is interacting with its environment during the given IoD.
Output Action	An action that writes or produces output data at an interface.
Output data	Data that is produced by a system.
PAR-Message	A PAR-Message (Positive Acknowledgment or Retransmission) is an error controlled transport service for the transmission of sporadic messages from a sender to a single receiver.
Payload of a Message	The bit pattern carried in the data field of the message.
Period	A cycle marked by a constant duration between the related states at the start and the end of the cycle.
Periodic Systems	A system where the temporal behaviour is structured into a sequence of periods.
Permission	Attributes that specify the access that subjects have to objects in the system.
Phase	A measure that increases linearly in each period from 0 degrees at the start until 360 degrees at the end of the period.
Phase alignment	The alignment of the phases between two periodic systems exhibiting the same period, such that a constant offset between the phases of the two systems is maintained.
Plaintext	Unencrypted data.
Precision	The precision of an ensemble of clocks denotes the maximum offset of (distance) respective ticks of the global time of any two clocks of the ensemble over the IoD. The precision is expressed in the number of ticks of the reference clock.
Prime mover	A human that interacts with the system according to his own goal.
Private Key	In an asymmetric cryptography scheme, the private or secret key of a key pair which must be kept confidential and is used to decrypt messages encrypted with the public key.
Problem	A perceived need to transform an initial state to a goal state.
Problem Elements	The initial state, the solution states, the acceptance test, the sub goals and the constraints that must be considered on the solution path.
Problem Framing	The process of describing and interpreting a problem within the problem domain (also the point of view, from which the problem is described).
Problem Solving	The process of finding a satisficing goal state. Problem solving encompasses (i) Analyzing and specifying the initial state (ii) Exploring the constraints in the problem domain (iii) Finding one or a set of satisficing

	goal states and (iv) Finding a solution path from the initial state to a selected goal state that observes the given constraints.
Problem Space	A state space that contains the initial state, the solution states and the identified constraints that the paths form the initial state to the solution state must satisfy.
Property	An instantiation of a characteristic quality.
Public Key	A cryptographic key that may be widely published and is used to enable the operation of an asymmetric cryptography scheme. This key is mathematically linked with a corresponding private key.
Public Key Cryptography	Cryptography that uses a public-private key pair for encryption and decryption.
Quality	The standard of something as measured against other things; the degree of excellence of something.
Quality of Experience/ Quality of Perception	A subjective measure of a user's experience/perception with a system and satisfaction for a system's quality.
Quality of Service	The ability of a system to meet certain requirements for different aspects of the system like performance, dependability, evolvability, security or cost; possibly expressed in terms of levels and quantitatively evaluated through metrics.
Raw data	The primary bit pattern that is produced by a sensor system.
Read/Write (RW) paradigm	At the sender the communication system reads the contents of the message from a message variable and at the receiver the communication system writes the arriving message into a message variable, overwriting the old content of the message variable.
Real-time system	A computer system for which the correct results must be produced within time constraints.
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.
Receive Instant	The instant when the last bit of a message arrives at the receiver.
Reconfigurability	The ability to repeatedly change and rearrange the components of an entity in a cost-effective way.
Reference clock	A hypothetical clock of very fine granularity, the state of which is in agreement with TAI.
Reference Monitor	A reference monitor represents the mechanism that implements the access control model. A reference monitor is defined as: An access control concept that refers to an abstract machine that mediates all accesses to objects by subjects.
Refined data	Data that has been created by a purposeful process from the raw data to simplify the explanation of the data in a given context.
Reliability	Continuity of service.
Relied Upon Message Interface (RUMI)	A message interface for the exchange of information among two or more CSs that establishes a well-defined boundary between the CSs that forms part of a backbone of an SoS system architecture.

Repeatability	Condition of measurement, out of a set of conditions that includes the same measurement procedure, same operators, same measuring system, same operating conditions and same location, and replicate measurements on the same or similar objects over a short period of time [5].
Requirement	A statement that identifies a necessary attribute, capability, characteristic, or quality of a system.
Resilience	Persistence of service delivery that can justifiably be trusted, when facing changes. Changes here may refer to unexpected failures, attacks or accidents (e.g., disasters). Changes may also refer to increased loads. Resilience thus deals with conditions that are outside the design envelope whereas other dependability metrics deal with conditions within the design envelope [3].
Resultant phenomenon	A phenomenon at the macro-level is resultant if it can be reduced to a sum of phenomena at the micro-level.
Risk	A measure of the extent to which an organization is threatened by a potential circumstance or event, and typically a function of 1) the adverse impacts that would arise if the circumstance or event occurs; and 2) the likelihood of occurrence.
Robust architecture	Architecture that performs sufficiently well under a variety of possible future developments.
Robustness	Dependability with respect to external faults.
Role player	A human that acts according to a given script during the execution of a system and could be replaced in principle by a cyber-physical system.
Safety	The absence of catastrophic consequences on the user(s) and on the environment.
Sampling	The observation of the value of relevant state variables at selected observation instants.
Scenario	A scenario is a projected or imagined sequence of events describing what could possibly happen in the future (or have happened in the past).
Scenario pruning	Scenario pruning is the process of discarding scenarios that include incorrect or unlikely information.
Scenario updating	Scenario updating addresses the problem of how to deal with newly available information, not present in the generated set of scenarios.
Scenario-Based Reasoning (SBR)	Systematic approach to generate, evaluate and manage different scenarios in a given context.
Second	An internationally standardized time measurement unit where the duration of a second is defined as 9 192 631 770 periods of oscillation of a specified transition of the Cesium atom 133.
Security	The composition of confidentiality, integrity, and availability; security requires in effect the concurrent existence of availability for authorized actions only, confidentiality, and integrity (with “improper” meaning “unauthorized”).
Security Level	Specification of the level of security to be achieved through the establishment and maintenance of protective measures.
Security Policy	Given identified subjects and objects, there must be a set of rules that are used by the system to determine whether a given subject can be permitted

	to gain access to a specific object. This is called the security policy.
Semantic Specification	The specification that explains the meaning of the named syntactic units.
Send Instant	The instant when the first bit of a message leaves the sender.
Service	The intended behaviour of a system.
Situation assessment	Situation assessment is the process of achieving, acquiring or maintaining situation awareness.
Situation awareness	The perception of the context in a given situation, the comprehension of their meaning and the projection of their status in the near future.
Situational Characteristics	Representation of the context of the problem that anchors the problem in the real world and elaborates the constraints.
Solution Path/Plan	A path of intermediate states from the initial state to the goal state, considering the given constraints.
Solution Set	The set of states that are considered a solution to the problem at hand.
SoS inference processes	Processes that map observations about the SoS to estimates about states that are relevant for decision making or planning but cannot be directly observed.
Sparse Events	Events that occur in the active interval of the sparse time.
Sparse Time	A time-base in a distributed computer system where the physical time is partitioned into an infinite sequence of active and passive intervals.
Sphere of Control (SoC)	The sphere of control of an entity during an IoD is defined by the set of entities that are under the control of the system.
Start signal	An event that causes the start of an action.
State	The state of a system at a given instant is the totality of the information from the past that can have an influence on the future behaviour of a system.
State Space	The state space of a system is formed by the totality of all state variables at that instant.
State Variable	A variable that holds information about the state.
Statefull action	An action that reads, consumes, writes or produces state.
Statefull System	A system that contains state at a considered level of abstraction.
Stateless Action	An action that produces output on the basis of input only and does not read, consume, write or produce state.
Stateless System	A system that does not contain state at a considered level of abstraction.
Static Attribute	An attribute of a property that does not change its value during the IoD.
Strategy Problem	A problem to devise a strategy or conceive a design that satisfies an abstract goal state under a myriad of explicit and implicit constraints.
Strong Emergence	An emergent phenomenon that is observed at the macro level is strongly emergent if, after a careful analysis of the emergent phenomenon, no trans-ordinal law that explains the appearance of the emergent phenomenon at the macro level out of the properties and interactions of the parts at the adjacent micro level is known (at least at present).
Structural	Representation of the abstract structure of the problem, focusing on the

Characteristics	problem elements and their interactions.
Sub Goal	A significant intermediate state on the solution path.
Subject	An active user, process, or device that causes information to flow among objects or changes the system state.
Subsystem	A subordinate system that is a part of an encompassing system.
Supervenience	The principle of Supervenience states that (Sup i) a given emerging phenomenon at the macro level can emerge out of many different arrangements or interactions of the parts at the micro-level while (Sup ii) a difference in the emerging phenomena at the macro level requires a difference in the arrangements or the interactions of the parts at the micro level.
Symmetric Cryptography	Cryptography using the same key for both encryption and decryption.
Symmetric Key	A cryptographic key that is used to perform both encryption and decryption.
Syntactic Specification	The specification that explains how the data field of a message is structured into syntactic units and assigns names to these syntactic units.
System	An entity that is capable of interacting with its environment and may be sensitive to the progression of time.
System Architecture	The blueprint of a design that establishes the overall structure, the major building blocks and the interfaces between these major building blocks and the environment.
System Boundary	A dividing line between two systems or between a system and its environment.
System effectiveness	The system's behaviour as compared to the desired behaviour.
System efficiency	The amount of resources the system needs to act in its environment.
System outage	The interval during which the system has failed, i.e. where the behaviour of the system deviated from its intended behaviour.
System performance	The combination of system effectiveness and system efficiency.
System resources	Renewable or consumable goods used to achieve a certain goal. E.g., a CPU, CPU-time, electricity.
System restoration	The transition from system failure to intended system behaviour.
System-of-Systems (SoS)	An SoS is an integration of a finite number of constituent systems (CS) which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal.
Temporal order	The temporal order of events is the order of events on the time line.
Testability	Property of a system to support testing.
Thing	A physical entity that has an identifiable existence in the physical world.
Threat	Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, individuals, or other organizations through a system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.

Tick	The event that increments the register is called the tick of the clock.
Timeline	A dense line denoting the independent progression of physical time from the past to the future.
Timestamp (of an event)	The timestamp of an event is the state of a selected clock at the instant of event occurrence.
Time-triggered (TT) Action	An action where the start signal is derived from the progression of time.
Tool	A tool is any item that can be used to achieve a goal.
Transaction	A related sequence of computational actions and communication actions.
Transaction Activity Interval	The interval between the start signal and the end signal of a transaction.
Trans-ordinal Law	A Law that explains the emergence of the whole and the new phenomena at the macro-level out of the properties and interactions of the parts at the lower adjacent micro-level.
Transport Duration	The duration between the send instant and the receive instant.
Trusted	A trusted system or component is one whose failure can break the security policy.
Trusted Computing Base	The set of components (hardware, software, human,…) whose correct functioning is sufficient to ensure that the security policy is enforced, or, more vividly, whose failure could cause a breach of the security policy.
Trustworthy	A system or component that will not fail with respect to the security policy.
TT-Message	A TT-Message (Time-Triggered) is an error controlled transport service for the transmission of periodic messages from a sender to many receivers.
Universe of Discourse (UoD)	The Universe of Discourse comprises the set of entities and the relations among the entities that are of interest when modeling the selected view of the world.
Unmanaged SoS evolution	Ongoing modification of the SoS that occurs as a result of ongoing changes in (some of) its CSs.
Value	An element of the admissible value set of an attribute.
Variable	A tuple consisting of data and a name, where the name points to the explanation of the data.
Virtual SoS	Lack of central purpose and central alignment.
Vulnerability	Weakness in a system, system security procedures, internal controls, or implementation that could be exploited by a threat.
Weak emergence	An emergent phenomenon that is observed at a macro level is weakly emergent if a trans-ordinal law that explains the occurrence of the emergent phenomenon at the macro level out of the properties and interactions of the parts at the adjacent micro level is known (or has been formulated post facto).
Well-structured Problem	A problem, where the initial state, the solution states, the acceptance test, and the constraints are well defined.
Worst Case Execution Time (WCET)	The worst case data independent execution time required to execute an action on a given computer.