ETSI GS ISI 006 V1.1.1 (2019-02)

Information Security Indicators (ISI); An ISI-driven Measurement and Event Management Architecture (IMA) and CSlang - A common ISI Semantics Specification Language

Disclaimer

The present document has been produced and approved by the Information Security Indicators (ISI) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGS/ISI-006

Keywords
cyber-defence, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Information Security Indicators (ISI).

The present document is included in a series of 9 ISI 00N specifications. These 9 specifications are the following (see figure 0 summarizing how the various concept is involved in event detection and interactions between all parts):

- ETSI GS ISI 001-1 [1] addressing (together with its associated guide ETSI GS ISI 001-2 [2]) information security indicators, meant to measure application and effectiveness of preventative measures.
- ETSI GS ISI 002 [3] addressing the underlying event classification model and the associated taxonomy.
- ETSI GS ISI 003 [i.1] addressing the key issue of assessing an organization's maturity level regarding overall event detection (technology/process/ people) in order to weigh event detection results.
- ETSI GS ISI 004 [i.2] addressing demonstration through examples how to produce indicators and how to detect the related events with various means and methods (with a classification of the main categories of use cases/symptoms).
- ETSI GS ISI 005 [i.3] addressing ways to produce security events and to test the effectiveness of existing detection means within organization (for major types of events), which is a more detailed and a more case by case approach than ETSI GS ISI 003 one [i.1] and which can therefore complement it.
- **ETSI GS ISI 006 (the present document) addressing another engineering part of the series, complementing ETSI GS ISI 004 [i.2] and focusing on the design of a cybersecurity language to model threat intelligence information and enable detection tools interoperability.**
- ETSI GS ISI 007 [i.6] addressing comprehensive guidelines to build and operate a secured SOC, especially regarding the architectural aspects, in a context where SOCs are often real control towers within organizations.
- ETSI GS ISI 008 [i.7] addressing and explaining how to make SIEM a whole approach which is truly integrated within an overall organization-wide and not only IT-oriented cyber defence.

Figure 0 summarizes the various concepts involved in event detection and the interactions between the specifications.

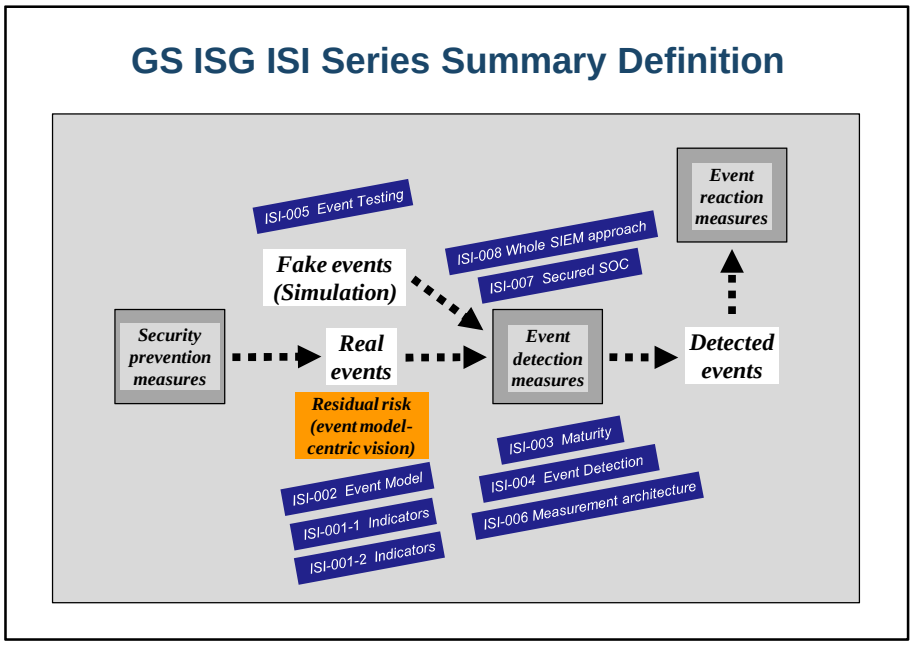

Figure 0: Positioning the 9 GS ISI against the 3 main security measures

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

The present document proposes an ISI Measurement Architecture (IMA) for the management of security events captured and contained by the *ISI Data Lake (IDL)* and which comprises raw data enriched by methods derived from ML Algorithms of the AI domain.

By means of the IDL sets of raw data should be typed, categorized and enriched in a unique manner for which formal Set and Graph Manipulation (S/G M) Theories and Techniques are applied. The ML-based classification mechanism uses a-priori learned information of a so-called *ISI-type matrix* containing the tuple pairs of ISI query tuple and the associated typed target tuple.

The dynamics of automation and control systems is modelled by *data<n-tuple>space* with the basic operations of *publishing, subscribing, etc. in order to manage ISI events* (i.e. formally events are graph edges of the intended semantics) that occur in Industrial Automation and Control [i.9] or other Ultra Large-Scale Systems [i.20]. The *ISI Data Lake (IDL)* functions as an *asynchronous memory* managing multiple security events at same time.

The compound IMA/IDL approach of the present document is based on theories of manipulation of sets and graphs combined with ML algorithms where appropriate. The latter applies pattern recognition measures for the purpose of enriching, i.e. filtering the raw ISI data representations of <n-tuples> from the IDL.

The notation *CSlang* is given in an operational style that supports the definition of Abstract Data Types (ADT), ML pattern matrices and ISI events. Since *CSlang* is intentionally not defined by a full formal grammar it is thus to be considered as a semiformal approach. Nevertheless it is intended to provide basic **schemes of comprehension** that deal with properties, e.g. the semantics of a concrete *ISI Signature* using types with variables, that are called sorts and operations with constraints that define the constraints respectively the invariants (axioms) of a type.

Cyber Security and Incident Event Management is an upcoming issue that is currently handled by several Standardization Committees and Industrial Specification Groups working on ISI classification and Cyber Security Evaluation. Other standardization activities such as Incident response management of the ISO/IEC SC27 have recently started. By this and other issues of complex security and safety evaluation and incident responses a need of more formality has been identified. Thus many project ToRs have raised the need to put more resources on approaches based on formal semantics and ontologies. Consequently the present document proposes an advanced standard of a common semantics specification approach that is able to fill the identified formality gap.

1 Scope

The present document provides a common interaction semantics model called ISI Measurement Architecture (IMA) based on formal approaches that are partially leaned from **Set and Graph Theories**, such as [i.8] and [i.16], etc. Graph Theory is the semantics background to reason by simulation, using appropriate tools. Between both, i.e. a foreground ontological specification and a background graph semantics pattern - a structure-preserving relationship should exist.

The given approach of the present document is meant among other things to support the incident reaction operation analysis performed by the staff of SOCs, in order to decide reasonably on observed security events and related measures. More specifically all stakeholders (CISOs, IT security managers, Designers, Programmers, etc.) get on hand a Common *ISI Semantics* Specification Language (called *CSlang*) which enables stakeholders to communicate in a common unique way to each other based on graph semantics. *CSlang* is designed to be a dialect of the **Common Logics(CL)** defined by the ISO/IEC SC32 Committee on Data Interchange in the international standard IS 24707 that share a uniform semantics based on *Traditional First Order Logics with Equality (TFOL)* according to [i.17] and [4].

The present document is structured as follows (after clauses 2 and 3 respectively dedicated to references and definition of terms, symbols and abbreviations):

- **Clause 4** describes models and methods of the ISI Measurement Architecture, including the challenge of transforming ISIs into knowledge about incidents.
- **Clause 5** invents advanced Common Logics (CL) concepts of the ISI Semantics Specification Language - CSlang.
- **Annex A** presents the Proof of Concepts (PoC) by aligning ontology specifications to graph specifications of the two levels of Semantics Approach.
- **Annex B** presents mathematical basic definitions of graph manipulation theory.
- **Annex C** documents authors and contributors.
- **Annex D** documents applied bibliography of semantic.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI GS ISI 001-1: "Information Security Indicators (ISI); Indicators (INC); Part 1: A full set of operational indicators for organizations to use to benchmark their security posture".

[2] ETSI GS ISI 001-2: "Information Security Indicators (ISI); Indicators (INC); Part 2: Guide to select operational indicators based on the full set given in part 1".

[3] ETSI GS ISI 002: "Information Security Indicators (ISI); Event Model A security event classification model and taxonomy".

[4] ISO/IEC 24707: "Information Technology - Common Logic - A Framework for a Family of Logic-based Languages".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GS ISI 003: "Information Security Indicators (ISI); Key Performance Security Indicators (KPSI) to evaluate the maturity of security event detection".

[i.2] ETSI GS ISI 004: "Information Security Indicators (ISI); Guidelines for event detection implementation".

[i.3] ETSI GS ISI 005: "Information Security Indicators (ISI); Guidelines for security event detection testing and assessment of detection effectiveness".

[i.4] ISO 27035-2:2016: "Information technology - Security techniques - Information security incident management -- Part 2: Guidelines to plan and prepare for incident response".

[i.5] Directive (EU) 2016/1148 of The European Parliament and of The Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union.

NOTE: Available at https://eur-lex.europa.eu/legal-content/EN/TXT/?toc=OJ:L:2016:194:TOC&uri=uriserv:OJ.L_.2016.194.01.0001.01.ENG.

[i.6] ETSI GS ISI 007: "Information Security Indicators (ISI); Guidelines for building and operating a secured Security Operations Center (SOC)".

[i.7] ETSI GS ISI 008: "Information Security Indicators (ISI); Description of an Overall Organization-wide Security Information and Event Management (SIEM) Approach".

[i.8] Peter D.Mosses(Ed.): "CASL Reference Manual", LNCS2960 Springer.

[i.9] IEC 62443-series: "Security for industrial automation and control systems".

[i.10] ISO/IEC 19086-2: "Cloud computing -- Service level agreement (SLA) framework -- Part 2: Metric model".

[i.11] OPC Foundation (07-19-2017): "OPC UA Companion Standard for Sercos".

NOTE: Available at https://opcfoundation.org.

[i.12] BSI.Bund: "Sicherheitsanalyse Open Platform Communications Unified Architecture (OPC UA)".

NOTE: Available at https://www.bsi.bund.de/DE/Publikationen/Studien/OPCUA/OPCUA_node.html.

[i.13] Wolfgang Ertel: "Grundkurs Künstliche Intelligenz - Computational Intelligence", 4. Auflage 2016, Springer Vieweg Verlag; ISBN 978-3-658-13548-5.

[i.14] Roberto Bruni, Andrea Corradini, Ugo Montanari, Universität Pisa, Italy: "Modelling a Service and Session Calculus with Hierarchical Graph Transformation".

[i.15] Claudia Ermel, Jens Richter, Jan deMeer: "Regelgestützte Modellierung von Anwender-Szenarien Kritischer Infrastrukturen für Analyse und Ausbildung" GI/ACM Regionalgruppe Berlin-Brandenburg, 22-11-2013.

[i.16] J.M.Spivey: "The Z-Notation - A Reference Model", C.A.R. Hoare Series Editor, Prentice Hall 1989.

[i.17] Jan de Meer et al.: "Introduction into Algebraic Specification based on the Language ACT ONE", Computer Networks - International Journal of Distributed Informatique, Vol.23, No.5, North Holland 1992.

[i.18] Axel Rennoch et al.: "Security Indicators Quick Reference Card".

NOTE: Available at https://cdn1.scrvt.com/fokus/e492943d2f291a76/4905070bb7ea30262ddf855393d14e21/SQC_Download_Etsi_isiQRC1.pdf.

[i.19] Dan Pilone: "UML2.0 - Taschenbibliothek", 2006 O'Reilly media.

[i.20] CMU SEI(June 2006) Pitsburg: "Ultra Large-scale Systems - The SW Challenge of the Future", Bill Pollak Chief Editor, created in performance of FG Contract FA8721-05-C-003, Linda Northorp ULS Study-lead.

NOTE: Available at https://insights.sei.cmu.edu/saturn/ultra-large-scale-systems/.

[i.21] Zohar Manna et al.: "The Logical Basis for Computer Programming - Vol. 1: Deductive Reasoning", 1985 Addison Wesley Publishing Inc.

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GS ISI 001-2 [2] and the following apply:

Abstract Data Type (ADT): specification of multiple sets of data, their properties and relationships among each other, in terms of sorts, operations and equations

Common Logics (CL): logic framework comprising syntax, higher order constructions and relations of a first-order modelling theory

data<n-tuple>space: structuring of the raw data space, called ISI Data Lake (IDL) by 'n-tuples', allowing processes to publish and subscribe upon

ISI Measurement Architecture (IMA): approach to enrich big dat sets, (i.e. ADTs) using methods from Graph Theory or Artificial Intelligence

OPC UA: M2M-communication-based Unified Architecture of the OPC Foundation

semantics: formal representation of system properties that provides formal reasoning on a mathematical level occasionally executable by modeling tools

3.2 Symbols

For the purposes of the present document, the symbols given in ETSI GS ISI 001-2 [2] apply.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS ISI 001-2 [2] and the following apply:

ADT	Abstract Data Type
CL	Common Logics

NOTE: See ISO/IEC 24707 [4].

CV	Continuous Variable
CSlang	Common ISI Semantics Specification Language
DOM	Data Object Model

GM Graph Manipulation (Tool/Theory)
HMI Human-Machine Interface

NOTE: See IEC 62443 [i.9].

IACS Industrial Automation and Control Systems
IDL ISI Data Lake
IEX Incident coming from EXternal sites
IMA ISI Measurement Architecture
ISI Information Security Indicators
ML Machine Learning (Tool)
SCADA Supervisory Control And Data Acquisition
SLA Service Level Agreement

NOTE: ISO/IEC 19086-2 [i.10] SLA Framework - p2 Metric Model.

STIX(TM) Structured Threat Information eXpression

NOTE: STIX 2.0 Draft http://stixproject.github.io/stix2.0/.

UUT Unit Under Test

NOTE: IEEE AutomaticTestMark-upLanguage.

XML eXtensible Markup Language

4 ISI Measurement Architecture - Models and Methods

4.1 The Challenge of transforming ISIs into Knowledge about Incidents

4.1.0 Introduction

The present document invents an advanced ISI Measurement Architecture (IMA) by a Big Data respectively ISI enrichment scheme. The process of Big Data Enrichment is intended to be supported by semantics-based tools from the shelf such as Machine Learning (ML), Graph Manipulation (GM), Ontology Specification (OS), Data Object (DO) Modelling, etc.

Firstly it is required to have a way of defining semantics for reasoning on ISIs and secondly, it is required to simulate designed ISI/IMA models. In case of IMA a compositional approach of Graph Manipulation together with Set Theories (i.e. Abstract Data Types) have been chosen to provide a semantics platform to represent distinctive IMA models.

A given formal model is set into relationship to an Industrial Automation and Control System (IACS) model that uses ontologies. If the relationship can be designed such that it is *structure-preserving* it is called a *homomorphism*. Checking homomorphism means to prove structural relationship between a given IACS ontological model with respect to its GM-based executable semantics model.

The anticipated *Communication Model* of IMA is based on an *data<n-tuple> space* i.e. a kind of platform that manages *ISI Events* such as incidents, measurements, data logging but also attacks and failures, etc. that are handled according to the principles of a *publish-subscribe communication paradigm* applicable to all components that exchange data<n-tuples>.

In figure 1 the '*Knowledge Pyramid*' respectively '*Knowledge Graph'*, is shown, of how to transform flat raw ISI related data into expert knowledge on security incidents. This approach is based on a so-called *Type Graph* (see next paragraph) that models e.g. an ISI Enrichment/Classification Process based on machine learning methods. When the so-called *learning matrix* - comprising typical pairs of queried and targeted incident patterns - has been sufficiently trained, it can be applied to the continuous classification process of unknown/untrained input patterns from an observed Industrial Automation and Control System (IACS). The unknown patterns stem from the basic entity nodes of the raw data level of the type graph in figure 1.

The anticipated ETSI GS ISI 006 (the present document) notation *CSlang - a Common (ISI Semantics) Specification Language* (as defined in clause 5) offers *semantic*, *static, dynamic and data typing specification* and modelling concepts. *Static* system properties are architectural design properties that are modelled by a so-called *Type Graph* representing architectural relations among components, devices, processes, stakeholders including humans. Dynamic system properties are behavioural design properties and are modelled by a so-called *Event Graph* representing communication relationships among data sources and targets that are interconnected by an 'ether' which is the so-called *ISI Data Lake (IDL)* that captures data representations as *data<n-tuples>*.

Finally strong Abstract Data Typing is achieved by means of a *many-sorted Algebr*a comprising data sets (SORTS), operations and functions (OPNS) on these sorts, typed variables, and *conditional equations (EQNS)*. A conditional equation is the algebraic specification equivalent of a system event that comprises an event head node and an event tail node, i.e. a *pair of ordered nodes* that is represented by an directed edge of the graph.

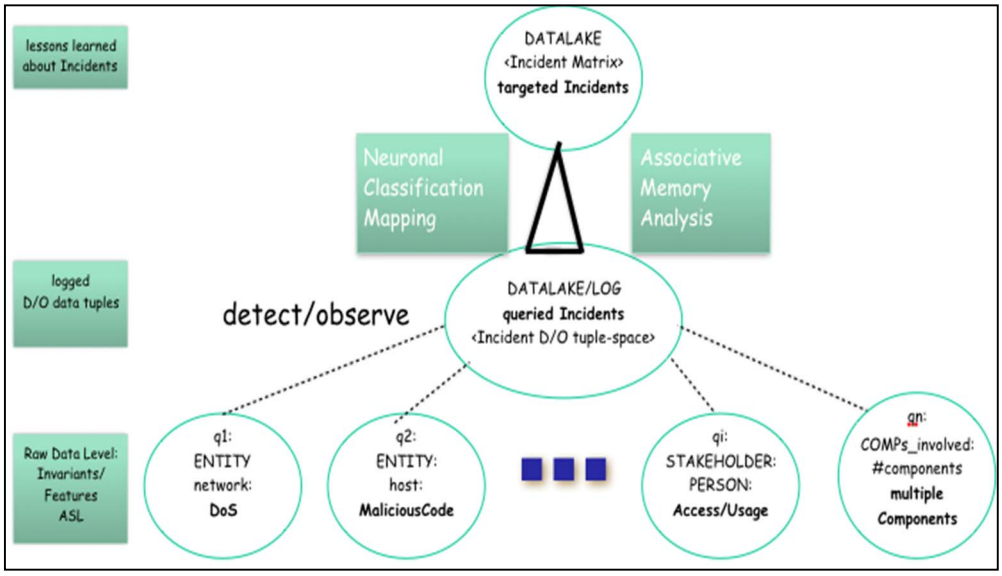

Figure 1: Type Graph representing the Knowledge Pyramid

4.1.1 Providing Upfront Indicators

To be efficient, a SOC of an IACSystem, will need appropriate *Security Detection Solutions* deployed in the right place in the Customer infrastructure, including the Cloud. In order to select the right Security Detection Solutions, the CISO needs to have a good understanding of its environment and to be able to quickly identify the types of security threats he needs to fight. By helping customers to identify upfront (see figure 1) types of incidents they are likely to face and map this with the catalogue of measures that will be relevant to cover these incidents, CISOs have a strong opportunity to quickly demonstrate, e.g. by means of the suggested tool box in the present document, the right decision to be taken.

4.1.2 The Human Factor

As mentioned in clause 4.1.0, whatever are the detection means and the security objectives, security events and security incidents can be managed by different solutions. But at the very end, the final qualification is still achieved by humans at the top of the knowledge pyramid of figure 1. This view provides human analysts with the capability to adjust ISI data classifications based on their own expertise and their knowledge of their customer environment. As a result, ISI generated in a specific customer context through specific services might not be relevant to another customer even if he is exposed to the same threats.

4.1.3 Out-of-Interest Frequency

For CISOs, security is a daily concern. Usually indicators managed by the SOC teams is on a monthly basis, i.e. CISOs will have less opportunity to monitor trends on real time basis and to take the right decisions in the right time. There is a need to look at managing Indicators more frequently which should also help SOC teams in identifying potential issues in the Real-Time (RT) detection capability. The latter is achieved by running GM Model in parallel to the SCADA which transforms any critical system state into a graph type and event model.

4.1.4 Continuous Measurement and *Excom* Reporting

Whether it is for justifying investigations or to check efficiency of security measures, CISOs have to regularly report using specific indicators. While ETSI GS ISI 001-1 [1] and ETSI GS ISI 001-2 [2] contain probably not the types of indicators that can directly be used for Excom reports, nevertheless they are the basement of any incident measurements of a company. Tangible data and a consistent approach are therefore required to produce ISIs mitigating the human factor and the RT issue.

4.2 The ISI Measurement Architecture (IMA)

4.2.1 The ISI Enrichment Approach

The following business requirements are drivers of the present document:

1) ISIs are to be used for **benchmarking customers systems** in a way using comparable inputs whatever is the type of customer system and whatever are the existing security detection solutions deployed in this, e.g. IACS infrastructure.

2) ISIs are to be managed and structured semantically based on a **formal model** with raw data enhancement strategies, using a standardized formal language, like *CSlang (of clause 5) of the present document*.

3) ISIs are to be classified using *Machine Learning (ML)* classification schemes which are embedded into a formal model.

The present document enables CISOs to produce classified indicators extending the ETSI GS ISI 001-1 [1] and ETSI GS ISI 001-2 [2] description scheme. The extension is achieved by referring to all raw security data published into the ISI Data Lake (IDL) of an IACSystem. By investigating into classified indicators of the IDL - the result may provide trends - i.e. simulations by the present document tool box can drive CISOs' decisions (what solutions CISOs should look at in priority? How can security detection solution more efficient to better detect, etc.). CISOs' decision support is achieved by the Graph Manipulation (GM) Capability in the present document tool box functioning as a model simulator allowing the simulation of possible impact predictions or forensic back-tracking to possible causes of observed incidents.

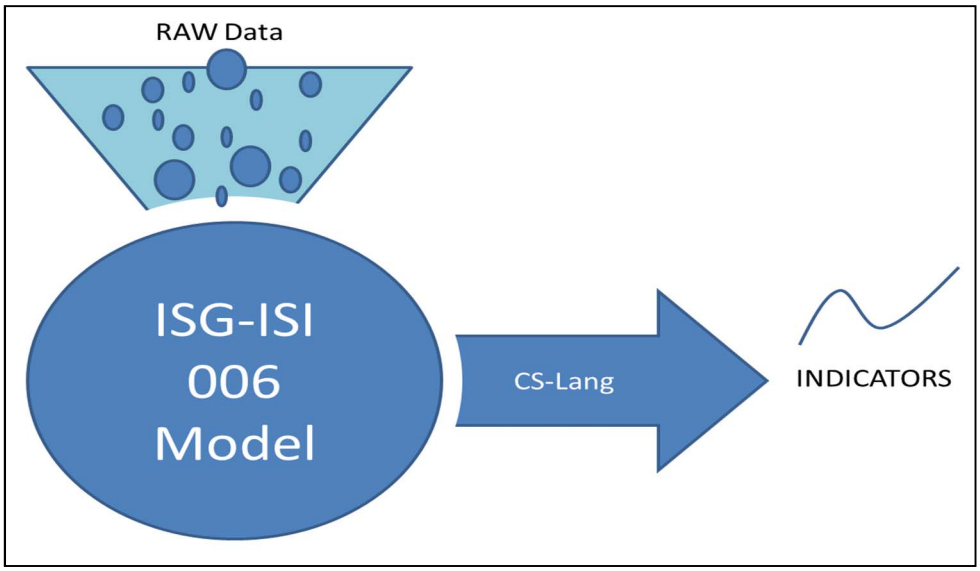

Figure 2: ISI Enrichment Process of the IMA (Data Lake)

In the following clauses the different views, called models, on the IMA are presented. These views explain the usability of the IMA with respect to the following modelling goals:

a) Common Language Concept, provides all necessary tools to express stakeholder roles, system use cases and activities, activity attributes, relationships (undirected graph edges) and associations (directed graph edges), Abstract Data Types with Variables (as part of graph vertices), etc.

b) Event Model, where an ISI-related event is modelled as an ordered pair of graph vertices and each vertex may contain typed system variables; the vertex in the system view usually represents a component; the data comprised by the event is a typed data<n-tuple>.

c) System Model, where the technical, security, privacy and safety issues and relationships among system components according the related industrial system standards of the IEC 62443 [i.9] are to be taken into account. For representation purposes ontologies comprising a certain universe of discourse are being applied which describes the context of ISI data application.

d) Enrichment Model, where AI /ML methods are used to enrich 'similar' data items into Abstract Data Types (classes) according to learned categorization information, i.e. a learning matrix used by Neural Networks.

4.2.2 The IMA - Common Language Approach

CSlang - Common (ISI Semantics) Specification Language is an approach for ISI Semantics Analysis of industrial automation and control processes providing Big Data Enrichment. *CSlang* is a semiformal approach developed to provide semantics to Industrial Automation and Control Systems (IACSystem definition according to IEC 62443 [i.9]) for the purpose of proving interoperability among heterogeneous machines, factories and humans.

CSlang is called semiformal because the language is not entirely based on formal grammars. *CSlang* allows the integration of existing Formal Description Techniques and Tools from the shelf such as the Graph Manipulation Tools, Model Checkers, Matlab, SCADA, AutomationML, WSDL, etc. provided there is a definition of an appropriate relationship between the formal semantics model and the technical IACS model. The latter model is represented by an ontology.

To represent semantics of a *Specifications in CSlang* the Graph Manipulation Theory has been chosen. The semantics of any specification is represented on two levels of semantics, i.e. the *1st level of semantics* representing applied Best Practices that are occasionally supported by tools such as 'SCADA"AutomationML', STIX, etc. and the *2nd level of semantics* that represents the formal Graph Manipulation Model being structural equivalent to the best practice 'SCADA' model, for example.

It is important to notice that all tools referred to can be taken from the shelf which allows everybody without sophisticated skills to check and prove the following system capabilities and properties:

1) Safe and efficient Security Information Event Management operating on data<n-tuples> of the so-called ISI Data Lake, respectively the Communication Ether, offering a common platform to all communicating IACS components and stakeholders.

2) Robust Machine Learning Processes (e.g. Associative Memory Mapping) to classify/evaluate raw ISI (big) data.

3) Available strong Abstract Data Typing describing complex composite *ISI data<n-tuples> and variables* by means of many-sorted term algebras.

4) Evaluation of Graph Manipulation Semantics to analyse, synthesize and predict IACS model behaviour.

Abstract Data Types (ADT) - mathematically spoken Algebras - build one of the formal language fundamentals of *CSlang*. The current approach of industrial component specification of *OPC UA* - *Open Platform Communication Unified Architecture* [i.11] and [i.12] is - formally seen - an ADT; hence OPC UA components are implemented as an ADT in *CSlang*.

ADTs comprise data sets, called SORTS to capture typed data; Relations between sorts, called OPNS are used to generate or derive data; and rules, called EQNS, that constrain the classification of data of a certain SORT. (Notice that there is not a clear distinction between **sorts and types**). However usually a sort denotes to a set characterized by certain properties, whereas a type obeys one or more sets and may have relationships between these sets. Thus a data type has the semantics of a multi-sorted algebra whereas a sort has the properties of a set of comparable elements.

The axiomatic approach of an ADT is an abstraction from any kind of language-dependent data representation. The axioms of an ADT can easily be derived from considering the data type properties, e.g. take a Boolean data type with non-canonical expressions; thus by applying the axioms of a model, it should be ensured that two or more expressions of a term equation never can be calculated to the contradiction (*true==false*); otherwise the checked model has a defect.

A combined ADT/GM approach compared to the OPC UA approach offers in a similar way the definition of operations of various data types and attributes that specify *subscription properties* like access conditions, access tracking conditions (e.g. graph manipulation sequences) etc., but also *publication properties* for alarming, state change of certain data types or system situations. Since the UPC UA model is event-based whose data to be published or subscribed are defined as an *data<n-tuple>* comprising:

<event-type time-of-publication severity-level-of-security <data-entry1…n> >.

Notice in the expression above, a blank marks the separator between descriptive elements of the <n-tuple>.

It should also be said that all elements of the event sample from above should all be typed. Thus the following ADT specification should proceed any event publication/subscription:

SPEC ISI_Data-Tuple_Sample IS

SORTS EVENT-ID, TIME-STAMPS, SEVERITY-LEVELS, DATA-ENTRY1,…,n; ….

OPNS …; EQNS …; ENDSPEC

The sorts above should contain all constants and generated values which are applied in the event sample. The defined sorts can also have variables.

It is further worth to be noticed that UPC UA calls this event data type specification to be *'a node'* similar to the node of a *CSlang* graph that is called a *vertex*. However graph theory is semantically more powerful since it allows to combine 2 nodes or 2 vertices to be related to each other by an ordering represented as a directed or undirected edge. Whereas an undirected edge represents the static architectural properties and a directed edge represents the dynamic event properties of the modelled behaviour. In that way, a directed edge symbolizes the 'flow of energy' i.e. the 'information that flows' from one node to the subsequent one. Hence tracking can easily be modelled and recorded. That allows for example, the modelling of dynamically changing information issued by events, e.g. weather conditions having impact on the state of a smart grid, i.e. the amount of energy being generated under volatile conditions and at same time the energy being consumed under almost static conditions, to be controlled during a period of time, like a day.

4.2.3 The IMA Event Model

An IACSystem comprises three basic components which are connected to each other by signalling paths and by flow-of-resources paths. Flows of resources are modelled as streams that may comprise physical things such as containers, energy, liquids, wind, sun irradiation, etc., but also data package flows. Flows of resources supply the **system assets,** e.g. an industrial production system**,** with any kind of required energy or goods. With respect to security and safety the assets should be kept under permanent control in order to avoid destruction, loss or malfunctioning.

The signalling and resource flow paths both are characterized by events that indicate a change of state, i.e. information is generated. To issue information has an addressee and thus has a direction which is marked by arcs, respectively edges of an **Event Graph** which is an extension of the Type Graph. Notice an edge modelling an event occurrence is defined of a directed pair of nodes, i.e. from the issuing (head) node to the target (tail) node.

ISI data capturing could comprise big data flows of instances of certain **CSlang Abstract Data Types (ADT)**. Data capturing may last for long or short periods of time and usually occur in smart applications, like finance, logistics, manufacturing, smart cities, interconnected things, etc. Streams of data packages of a distinct type are modelled as flows of discrete data records represented as **<n-tuples>** that are published to the IDL. So, from an event management point of view, it is to be distinguished on handling a full stream as an event or just a single data package.

For certain Control and Monitoring task - it would make sense not to compile massive data type sets first and then to put massive computing power for analysing purposes - but to check (in Real Time) for **significant changes** in the observed flow of data or resources. A single change of state in the flow of data may provide characteristic information about a component's behaviour, bad or good. The learned information about should be captured instead of collecting massive data sets itself. Clearly it is information derived from events subscribed at the IDL; i.e. the collection of meta-data derived from an observed **CSlang Continuous Variable** (see clause 5).

4.2.4 The IMA - Enrichment Model

The ISI Measurement Architecture (IMA) underlying *communication ether* is denominated to as the **ISI Data Lake (IDL).** The IDL is organized as an hierarchical model managing data items/instances represented as <n-tuples>. Thus the IDL is understood as a **repository** for all <*n-tuples*> being published or subscribed during life-time of a system. The publisher, i.e. the component which generates measurement data does not need to send them to distinctive receivers but simply publishes them as *data*<*n-tuples*> into the IDL. Since all data<n-tuples> are typed by a CSlang ADT a consumer subscribes to the IDL to specific ADTs he is interested in without being interconnected to the data sources.

The **event management** of the ISI Data Lake is said to manage the time gap between pairs of IDL events of publication and subscription which in turn can be modelled as a new more abstract event. Event management applies the IDL repository's capabilities of event observers or event notifiers to each event type to be managed. Since all events happen asynchronously, subscription could even occur before publication of the same ADT and vice versa. The final synchronization is achieved by the event management inside the IDL.

The **enrichment process** is part of the IDL and is explained in detail in clause 4.3.3 (PoC-ML). However enrichment is understood as the process of guessing for unknown data patterns the appropriate classes they may belong to. The process of classification is achieved by a machine learning algorithm based on neural networks as it is exemplified in the PoC-ML. In CSlang a class is specified by a multi-sorted ADT. Classification is a fine tool of a tremendous reduction of ISI measurement data because all items of a class are handled in a similar unique way.

The **Event Graph of Classification** comprises a component with the matrices of learned pairs of head-tail nodes capturing the well-known relationship of ISIs to vulnerability/incident classes, the SOC needs to react on. The matrix is a many-to-one matrix, i.e. many input patterns (the ISI queries) are classified into one output pattern (the class target).

On a higher Human-Machine Interface, i.e. SCADA, the classified ISI queries can easily be handled and simulated by exploring an Event Graph that supports cause-impact analysis of classified incidents.

4.3 The PoC Use Cases

4.3.1 The General Tooling Property

In Annex A there are two figures about the semantics notion of homomorphy.

Homomorphy is the property of '**structure preserving**' relationship between different representations of the same system model.

In figure A.1 there are two levels of semantics, i.e. one level is an ontology-based representation the other one is graph-based. Hence for this configuration structure preserving means that a state change in the ontological representation, e.g. an HMI-interaction of the model, should also occur as comparable state change in the graph representation. More formally it can be expressed by the equality of the two sequential (seq operator: 'o') mappings: v o f == g o v', whereas v and v' are the state changes of the graph or ontology representations and f and g are the relationships among the semantics objects between the graph and the ontology. The relationships f and g should be matters of standardization because there are no general rules of guiding or even constraining the mapping.

In figure A.2 there are two homomorphic representations of the ontology of an electrical circuit and its structure-preserving model of a graph representation. Whereas the electrical circuit interconnects the tree devices of a DC source G1, a switch S1 and an electrical consumer P1 with the interfaces (1-11), (12-1), (2-2) adopting the device-internal electrical conditions to each other's device by maintaining the electrical invariance of the maximum electrical current, i.e. I = 0,5A for each device.

Similar the graph obeying vertices and edges related to devices and interfaces. At the interfaces interactions may happen, provided the internal constraints of the interacting devices allow the current $i \leq 0{,}5A$ to flow. Hence there are three events (i_1, i_2, i_3) that correspond to the three interfaces of the ontology.

NOTE: The same homomorphy relationship does also exists between the UML2-based knowledge gaining methodology of CSlang (as explained in clause 5) and the guidance toward a structure-preserving construction of graph semantics.

4.3.2 PoC-GM, the Graph Manipulation (GM) Tool

By means of the GM tool graph manipulations are executed that make visible recorded (past) versus guessed (future) sequences of events. Obviously a GM Model is not the same as a ML/Neuron model; however their complementarities are reason enough to combine both, ML and GM biased models into a powerful semantics notion.

The structure of the ISI Data Lake (IDL), in which one can see that the IDL comprises the full range of data types from raw unstructured formats to sophisticated highly structured data type formats. The IDL plays the role of a central communication means, i.e. the platform (also called ether) where all data handled by *events* - i.e. *publishing* (writing instances of data types to the IDL), *subscribing* (searching for data types in the IDL) or *consuming* (removing data types from the IDL) - are defined as abstract data types that are represented as *data<n-tuples>* and managed in a unique way, i.e. by a global IMA approach.

Since an event is something that is observable and something that dynamically happens. The **observability** means there is a known event type modelled by the type graph and a **dynamically changing environment** making the event type happening.

Hence two types of graphs are to be constructed: the first type is the general, almost static **Type Graph** comprising the system architecture of components with their Data Types - which are represented by vertices, i.e. nodes interconnected by undirected edges. Such type graphs convert into event graphs when undirected edges become directed edges, i.e. when events are going to happen dynamically, i.e. the environment is changing.

For example, if a system component state - in figure A.2 *graphically* represented by a *vertex* - is defined by a set of *typed variables,* then a system component state change - graphically represented by an *edge* - that interconnects a pair of *vertices (j k).* The pair of vertices models an *event* that transforms the system component state of *vertex(j)* into another one, the next state of *vertex(k)*; whereas *vertex(j)* is called '**event head**' and *vertex(k)* is called'**event tail**'. Hence event head and tail are interconnected by a directed edge *edge(j,k)* that can be simulated (i.e. events to be added, removed, conditioned switching conditions, etc.) by a suitable Graph Manipulation Tool from the shelf.

Since the latter graph obeys *directed edges* representing *events* it is called **Event Graph** - the former graph obeys *undirected edges* representing *architectural relationships*, it is thus called **Type Graph** to make the difference.

PoC-GM: Figure A.2 shows an example of an electrical network that should be verified: Two drawings are given, i.e. the left one is a real electrical switching network that interconnects three devices, a Voltage source G_1, an electrical switch S_1 and an electrical lamp P_1. The right drawing is an Event Graph derived from the electrical network with three nodes representing the three devices of the switching network from the left.

If the switch of the electrical network is closed three immediate events as represented in the Event Graph by (i_1, i_2, i_3) are going to happen, i.e. the pairwise node variables of the event head (lhs) and event tail (rhs):

'event_head(Vars)=>event_tail(Vars)'

fulfil the *switching conditions* and thus issue an *event*. In the real electrical network current is flowing.

NOTE: During all possible events never, the invariant condition is allowed to be violated, e.g. this however would be the case if the electrical lamp is replaced by a short-cut.

4.3.3 PoC-ML, the Machine-Learning (ML) Tool

Basically the Machine Learning Tool mitigates partially the human factor by providing a common methodology to categorize/classify multiple big data sets in order to prepare human decisions.

The ISI Enrichment Process uses tools and methods from the domain of Computational/Artificial Intelligence, i.e. Cognitive Systems (CS) (see note 1) that are able to recognize patterns by means of Machine Learning(ML) Algorithms. ML is an approach that is useful for categorizing big data sets. ML based on a learned matrix W comprising pairs of patterns (query target) that are combined into an a*ssociative memory* synonymously denoted to as *neuron* [i.13] or a combination of neurons of a neuronal network. A neuron represents a so-called activation function that is applied to the sum of all weighted values of an input pattern. In case of binary patterns the weighted sum turns into a logical OR. The *binary learned (weight) matrix* W is **similarity-preserving**, i.e. any query input pattern q is related to a similar target output patter t according to W, whereas the latter output pattern is the found class indicator.

NOTE 1: It is important to make a distinction between Cognitive and Autonomous Systems (CS, AS) since both provide a kind of memorizing control mechanism, i.e. CS uses the learning matrix Wij(Neuron) to memorize similar patterns and, AS uses feed-back to memorize similar system states (e.g. all former events that have led to the current system state).

A simple view on an associative memory is like a table of pairs of values, in which the search and update of a value is achieved by keys. The table entry pair (key value) is compared to the neuron pairs of (q^p t^p) which represents the learned p^{th} entry of the (e.g. incident) matrix W, whereas the query pattern is considered as the key and the target pattern as the value *learned* from applying W, e.g. the identifier of a class of ISIs.

NOTE 2: An ML Proof of Concept is explained in detail in Annex B.

5 The Common ISI Semantics Specification Language

5.1 Introduction into CSlang - A Common Language

CSlang is not intended to provide a full formal grammar and can thus be classified as a semiformal language. Nevertheless *CSlang* is intended to provide **schemas of comprehension** in order to express dynamic and static properties and characteristics of systems compared to the standardization of Industrial Automation and Control Systems, i.e. IEC 62443 [i.9] IAC System safety and security properties.

The specification of system properties - hence semantics - is achieved by the invention of **graph and set theories**. A system graph models the system capabilities(statics) and behaviour(dynamics) by benefitting from two graph representations, i.e. the type graph and the event graph. The Abstract Data Types (ADT) based on set theory, model the data<n-tuple>space and classification schemes for ISI patterns. Classification is achieved by applying two *CSlang* tools respectively schemes, i.e. the specification of an *ISI Signature* that uses data types with variables (sorts), and an ISI enrichment scheme, that uses learned patterns mapping unknown ISI patterns into associated ISI classes.

Operational semantics of e.g. an IAC System can thus be represented by a formal (knowledge) graph representing the static architecture and the dynamic behaviour of the modelled system. Since the architecture is static with respect to its capabilities, the graph is static too, i.e. without events formed by directed graph edges. When the system comes into operation events and its observations about activities are going to happen that makes the graph dynamically changing i.e. the graph's static undirected edges are transforming into dynamic (also called switching) edges that are related to system events.

Whereas static and dynamic system properties are represented by sub-graphs, ISI data properties are modelled by 'many sorted types' represented as an Abstract Data Type with Variables. Thus an ADT contains the specification of the attributes, interfaces and the ISI data<n-tuples> to be communicated and which - by the designer - are aligned with the various graph nodes determining the system components obeying states.

The necessary information required to construct a semantics system graph is derived from a 5-step comprising method using selected basic UML2 stereotypes [i.19] such as:

1) the system together with the Use Case to be described; and

2) the system environment (both represented by UML2 rectangles);

3) the stakeholders and actors (represented by UML2 symbols or icons) applying the system capabilities at human-machine interfaces (HMI);

4) the roles actors may play (represented by UML2 relationships) at those HMIs;

5) relations and associations between interacting system components respectively graph nodes/vertices (represented by UML2 lines and dotted arrows).

At first, the information gained from applying the UML2-based method above, forms the informal knowledge base of the system which then is used to specify formally the system characteristics in terms of sorts, variables, axioms, invariants and basic or complex many-sorted Abstract Data Types (ADT) and finally the ADT specifications are to be integrated into the system graph.

Thus, according to figure 5, the type and event graph can be constructed, comprising nodes respectively vertices and directed or undirected edges. In figure 5 a sample is given of a classification process represented by a subgraph of classifying IEX.INT indicator data tuples. The ISI data<n-tuples> contain typed variables and constants of the so-called lefthand-side of the CSlang conditional equation respectively rule, that are transformed into the righthand-side of the CSlang rule, provided the defined thresholds of the intended semantics are achieved respectively not violated. Notice that the classification process may comprise several states of classification and thus comprises more than one step depicted by a sequence of directed graph edges (see clause 5.5) that form a classification process.

The many-step comprising classification process is typical when classification is based on neural networks (NN) which is explained in detail in the PoC of clause 4.3. Principally there is no difference in performing classification processes using accumulation of simple statistical values or, using well-known values (patterns) saved in a learning matrix W contained and maintained by a NN-subgraph and further the classification decision is taken on un-known incoming patterns being 'similar' to that class or not which depends from the learned matrix modelled by successive graph edges.

5.2 Data Property Specification Scheme

When a schema comprises a Signature S with an Expression E, according to [i.16], then it is written as: {S o E} with the meaning: the *scope of the typed variables,* declared in the specification S, comprises all subexpressions of E.

A property over a *Signature S* is characterized by all *Use Cases* in which it is valid. A Signature S comprises the declaration of all variables obeying a type. The typed variables provide the **vocabulary** of how to make statements about properties that are expressed as logical predicates (e.g. x:t<y:t of two variables x,y both of type t).

A predicate - i.e. a property over a signature - can also be expressed as well by equations with variables: 'E1=E2' respectively 'lefthand side => righthand side' obeying equality of meaning of E1 and E2, but not equality of the two expressions 'lhs' and 'rhs'. These equations are also called rewrite rules, because the 'lhs' can be replaced by 'rhs' provided the 'switching condition' of the conditional equation are fulfilled.

Equational Logic is useful when functions respectively Operations are involved, e.g. as it is demonstrated by the specification of the constrained Integer type below that comprises the Integer-Signature with SORTS and OPNS, and some equational expressions with variables (called EQNS) that constrain the set of Integers to set of positive numbers comprising zero.

Thus the basic CSlang semantic elements to express static properties of a system are as follows:

1) **Data Sets**, denominated SORTS, e.g. 'number', 'sensor_temperature °C';

2) **Relations** between Data Sets, i.e. Functions denominated OPERATIONS, e.g. *'+1', '-1'*;

3) **Constraining Properties, i.e. Invariants** on Data Sets, i.e. denominated by EQUATIONS;

   e.g. '-1(+1(number)) —> number' that classifies different representations of SORT number;

4) **Variables** are elements to describe general properties of data sets in an indefinite style. A variable is a placeholder for any expression of a SORT representing a set of data. Variables can be used in EQUATIONS, e.g. forall 'var' of 'sort' is -1(+1(var)) = var'.

In the signature part of the TYPE SPEC Example below there are the declarations (called SORTS) of the only *set type* NAT with the *variable* n:NAT, the *unary operation type* 'NAT—>NAT' and the *constant operation type* '0—>NAT', e.g.:

————————————-TYPE SPEC Integer Signature ————————————

SORTS n:NAT; //one sort 'NAT' with Variable 'n'//

OPNS +1,-1:NAT—>NAT, 0:—>NAT; //n-ary operations:+1,-1,0//

————————————TYPE SPEC Integer Semantics————————————

EQNS //equational semantics//

-1(+1(n)) = n,

+1(-1(n)) = if n==0 then +1(n) else n, //conditional equation//

+1(n) = 'n+1',

-1(0) = 0;

————————————————————ENDSPEC————————————————————

The type specification 'Integers' contains only natural numbers with zero, because its semantics part (i.e. the EQNS-part) restricts the defined operations (of the OPNS-part) not resulting into a negative number, even any operation is applicable. It is interesting to notice that this single **invariant constraint** is expressed by all four equation making the distinction between the two use cases of n = 0 or n ≥1.

5.3 Process Property Specification Scheme

In order to achieve the *ETSI ISI* specification scheme (as it is known from the ETSI GS ISI 001-1 [1] and ETSI GS ISI 001-2 [2]) that is based on statistical numbers, CSlang supports it by its notion of **event type**. An event is something that 'happens' with the meaning of information generation and transportation.

For example, assume a configuration of a temperature sensor device connected to an indicator computation component. The embedding environment of this configuration is the communication ether of IMA, the data<n-tuple>space.

The **compound CSlang semantics elements** to express dynamic properties of a system are as follows:

1) the IMA communication ether modelling the system's environment that supports the communication of data**<n-tuples>.** A data <n-tuple> is a many-sorted data type whose typed elements are sequentially enumerated. For example, the EVENT_TYPE respectively the GRAPH_EDGE <event-type:=SensorTemperature °C> generates the following data <n-tuple> with variables assignments that determine the 'happening', i.e. switching conditions of the event:

   data<n-tuple> Example:

      <event-type every-minute start-time end-time over/under-threshold>;

2) an **Event Type** is a small subgraph comprising two nodes with ***variables*** and a single edge represented by a conditional equation with the three typed expressions 'guard', 'lhs', 'rhs'; whereas 'guard' has the type 'binary' and the expression 'lhs' has the type derived from node 'event head' and expression 'rhs' has the resulting type derived from node 'event tail':

   forall 'var' of 'sort' is '[guard] lhs => rhs';

3) **process** is a complex system graph operating continuously on defined ADTs in order to achieve a certain goal, like:

   - to enrich ISI raw data sets such that, e.g. the meantime of temperature during a period of time can be computed;
   - to classify unknown patterns according to a learned matrix of weights based on Neuronal Networks;
   - to simulate any dynamic behaviour of autonomous control systems in terms of conditioned events, in terms of CSlang stereotypes shown in figure 3.

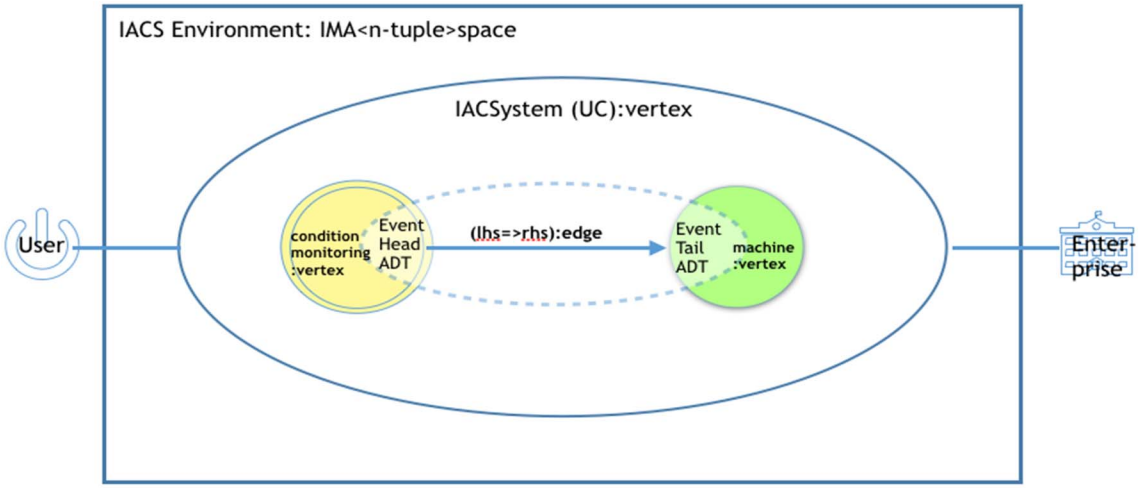

Figure 3: CSlang semantic elements (stereotypes)

Events of a process have the characteristics that they eventually happen. Events that have happened are semantically represented by a **graph manipulation**, i.e. a new directed graph edge is inserted into the existing type graph. The presupposition of the edge insertion is that the edge pertaining to the event should not yet be directed. By this graph manipulation a switching event becomes an ordered pair of graph vertices or, in other words a directed edge, whereas the first vertex is called '**event head**' and the second vertex is called '**event tail**'. The switching event is given as a rewrite-rule like 'lhs=>rhs' with lhs is the enabling event expression and 'rhs' is the resulting event expression. The figure above the basic and compound concepts of CSlang based on UML2 stereotypes are demonstrated that model an IACSystem (see figure 3).

5.4 Data Object Model Specification Scheme

An Abstract Data Type (ADT) can be considered as a formalization of a descriptive Data Object Model (DOM) that is usually specified in terms of XML, UML or STIX like languages. But with CSlang-type of notation those representations can be given a unique operational style of semantics.

Operational semantics means to make of a descriptive DOM a constructive DOM. For example a book with all its components is translated into (algebraic) sets, operations and constraints, i.e. the book is opened by the operation 'open' yielding some pages of the book which can be used by another operation 'read' yielding information from the open pages.

In the following sections the 'DOM book' is developed as an ADT: at first specify to specify all intended sorts of the DOM book comprising three declarations of different sorts are to be specified in the following manner:

NOTE 1: A 'blank' letter is used to separate the conceptual declaration in this simple example.

SORTS any_book some_pages information;

followed by the specification of all required operations to use the DOM book:

OPNS OPEN: any_book —> some_page

READ: some_pages —> information

WRITE: information some_page —> some_page;

The operations above determine also the **data type coercions**, e.g. from 'any_book' to 'some_page'. Coercion can also occur with more than one source types into one target type as it is demonstrated with the 'WRITE' operation with the semantics of writing some data of a simple type 'information' onto a complex data type 'some_page' with a state change after having finished the WRITE operation, i.e. the 'information' has finally be written onto 'some_page' contained in 'any_book'.

Of course there should also be some invariants not being violated when using or writing into the book. In the intended sense of the DOM book signature the three declared operations by a user can be applied at any sequence if there would be no constraints. Thus pages of the book cannot be read before the book has been opened. Or, empty pages do not deal with information since the pages are not filled with text and so on. In order to handle those invariants at first, it is necessary to declare some initial **constants** of the sorts 'any_book' and 'some_age' to be added to the signature above by null-ary operations, called constants:

CONST **empty**_book: => any_book

empty_page: => some_page

Secondly any element of the defined sorts is represented by a **variable of a sort** which is specified in the EQNS part of the Signature, e.g. the variable 'book' represents all element of the sort 'any_book':

EQNS FORALL **book**: any_book, **page**: some_page, **info**: information,

OFSORT some_page

OPEN ([**empty**_book]**book**) == **empty**_page

OPEN (['true']**book**} == **page;**

NOTE 2: In the above DOM book specification **conditional equations** are used by having applied a 'guard ['bool']OPNS' that can constrain any operation contained in the lhs of an equation. The guard should be able to 'switch', i.e. to be fulfilled, before rewriting of lhs => rhs terms is applicable.

However it should be differentiated between the notions of SORTS and VAR OFSORT whereas a sort is a set and contains all terms or expressions of that sort, but a variable ranges of all terms of the SORT: For example the variable 'info' may contain at different stages of processing different values, i.e. different terms of sort information, e.g. VAR info' and VAR info differentiated by the dash indicating sequential stages:

VAR **info'<—** READ(WRITE(**info** OPEN(**book**)));

The expression above has the semantics that one can only read an open book into which something can be written, i.e. 'write(info) onto pages of_a book'. However before writing the book should be checked to be empty or not. Thus there are minimum two valid models, since all defined invariants hold for an empty and non-empty book. Writing into an empty book however cannot be assumed thus it should be checked before!

When generating terms by operations all semantical invariants and constraints should be stated explicitly. For example a syntactic constraint is that of the type-arity of every operation. In the given example above there are the constraints of 2 unary and 1 binary operations and the coercion constraint of the real object variable 'book' with the declared sorts, e.g. 'read(page)' would be an invalid expression because it is only read on opened book!

Coercion is achieved with any term (of the ∑-Algebra) that is evaluated inside-out, i.e. some pages can only be read if the book has been able to be opened, i.e. there should be at least some pages compiled in the book. An empty book would not be able to be opened and will yield 'error' (which is another constant to be declared) und hence trying to read('error') will stop the rewriting, i.e. it cannot operationally be executed furthermore.

The operations 'open' and 'read' are written as functions by using the symbol '—>', that **coerces types**: 'any_book' into 'some_page' and 'some_page' into 'information'. The type coercion (arrow symbol) also symbolizes a state change of the DOM book, namely the events that happen correctly in the right sequence: at first 'open(book)' followed by 'write(page)' or 'read(page)' Thus type coercion checking is a kind of proving **correctness**!

So far there are only few expressions able to be built because other constructive operations such as 'write(info OPEN(book))—> some_page' or 'empty_book —>any_book' or 'empty_page—>some_page' have not been applied. By the 'write' operation the book can be written page per page starting with the empty book, i.e. the constant 'empty_book'.

5.5 ISI Measurement Specification Scheme

In the ISI sample of figure 4 'IEX.INT' (taken from the list of ISIs collated in [i.18], [1] and [2]) there are found for this selected class three ISI event types:

1) Intrusion attempts on external accessible servers.

2) Intrusion detection on external accessible servers.

3) Intrusion on internal accessible servers.

Each event type is semantically described by an **ordered pair of graph vertices. A**ccording to the definition of a graph, pairs of vertices define graph edges. Thus for each event type, (i=1,2,3) there should be a graph edge respectively a pair of nodes: $(INC_i\ MEA_i)|_{i=1,2,3}$ where the INC_i node represents a process that measures statistical parameters of possible incidents, and the MEAi node contains the resultant indicator values of the IEX.INT indicator in terms of the evaluation of the 4-tuple <f s d m> comprising the typed variables:

VAR f OFSORT frequency={f|low medium high significant keytoknow}

VAR s OFSORT severity={s|1.low 2.moderate 3.high 4.highest}

VAR d OFSORT detection_rate={d|1.very_difficult 2.difficult 3.easy}

VAR m OFSORT maturity_level={m|…}.

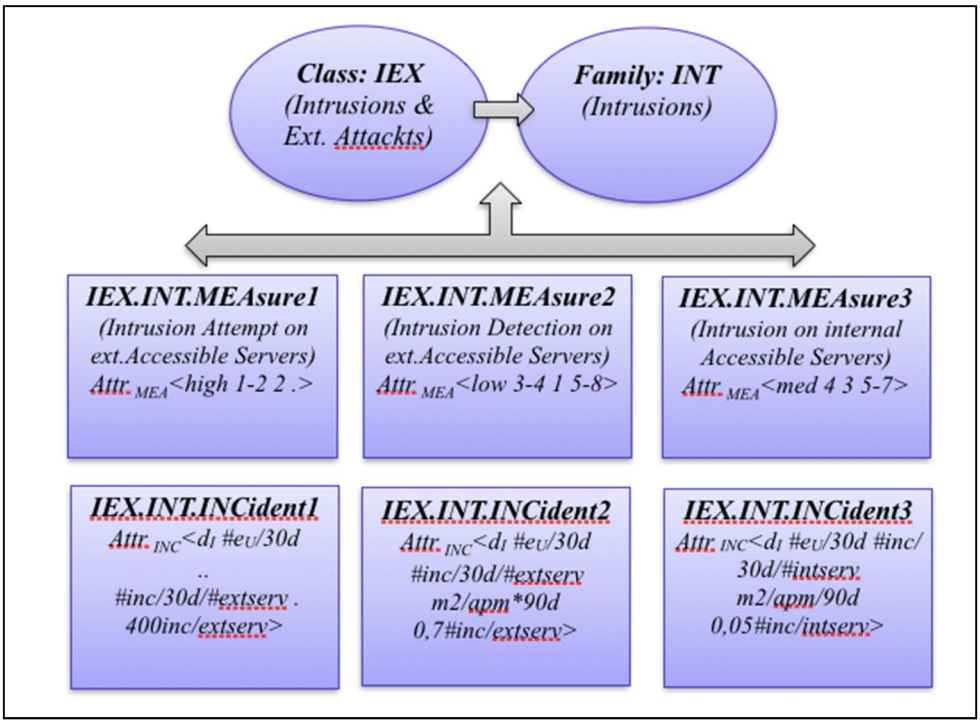

Figure 4: ISI Samples of ISI Indicator class IEX.INT

Since the semantical notions of an **ISI type graph are vertices and edges** and since edges represent relationships between pairs of vertices, edges model the dynamics of a system by means of events. In case of the ISI samples of figure 4, each ISI event is defined by: the INT.INC type event head node with the expected incident parameters of the INC.j<6-tuples> and the INT.MEA type event tail node with the final evaluation values of the MEA.i<4-tuples>.

The construction of the (knowledge) subgraph of the ISI class IEX.INT is depicted in figure 5.

NOTE: In the sample the specification of the IEX.INT type graph is in accordance with the specification style of the graph manipulation tool [i.15] used for the experimental simulation of a smart grid use case invented elsewhere.

An intended ISI classification scheme can be analysed by using the '*knowledge of the past(KoP)*' and the *'knowledge of the future(KoF)'* both gained from simulations. *KoP* means keeping the ISI subgraph and going backward looking for reasons why the current state is so secure or insecure; *KoF* means extending the ISI subgraph and going forward analysing the effect of assumptions in order to prepare decision-making. In both use cases knowledge is learned from the analysis of the ISI class IEX.INT modelled as subgraph.

The gained knowledge can be transformed into a so-called learned matrix W as it is described in section 4.3 of the PoC Use Cases. The measurements of the statistics processes of the node IEX.INT.INC can be 'weighted' individually by the three learned matrices of each of the three intrusion event types yielding **similarity-preserving** results (i.e. the weighted patterns contained in W) processed by the nodes $IEX.INT.MEA|_{i=1,2,3}$.

Thus the IEX.INT type subgraphs can be constructed yielding the edges: $IEX.INT.INC|_{i=1,2,3}$ *$W|_{i=1,2,3}$ —> $IEX.INT.MEA|_{i=1,2,3}$ (outlined in figure 5) which determine the event head nodes. The event tail nodes of this edge, i.e. the final classification of the related ISI class yields the best results taking into account the previously from simulations gained knowledge saved in W.

In CSlang processes represented by graph nodes sometimes are modelled by so-called **Continuous Variables (CV)** denoted to as X^{dot}, of the differential variable dx/dt. As other variables CSlang's continuous variables are as well strongly typed. It is appropriate to use CVs for accelerating **raw data**, to be collected by a continuous variable over a period of time and finally compiled into a set of statistical values, e.g. by the INC.j<6-tuples> CVs. The counter as an example of a continuous variable - denoted to by X^{dot} - that may operate on one or more other 'simple' variables $<X|x_1 x_2 \ldots x_n>$.

In figure 4 the **processes** respectively **continuous variables** are specified using three IEX.INT.INC(i=1,2,3) sets of attributes. For example the *process of counting* is either achieved by a counting rate or it is triggered by a deployed sensor measuring something more specific and which is attributed with characteristics of intrusion attempts or incidents observed over a period of time.

In order to compile an ISIndicator of the class IEX.INT based on MEA_3/INC_3 node attribute-tuples (see figure 5) having the meaning '*Successful Intrusion Detections on External Accessible Servers*' some knowledge should be taken into account experienced from pre-defined best practices of using MEA_3 tuples, e.g. *<f:med s:4 d:3 m:5-7>*. Of course intrusion detections should happen with a low frequency rate when one or more of the servers have subscribed to this measure e.g. for the purpose of compiling valid ISIndicator classes like IEX.INT during a period over 90 days. In that case the ISIndicator classifying incidents of the category (i=3) comprises the following attributes compiled into the $IEX.INT.INC_{i|(i=1,2,3)}$ data<n-tuple> of *CSlang*:

<IEX.INT.INC_3 dateOfIncident

#intrusionAttempts/30d

#incidentsObserved/#servers/30d

#averageIncidents/90d

5%#incidents/#internals>.

After the INC measurement period - in this case of 90 days - the ISI information of the measured $IEX.INT.MEA_3$ is published through the IMA data<n-tuple>space (the type graph communication environment) to which in turn other stakeholders can subscribe and retrieve information about classified incidents.

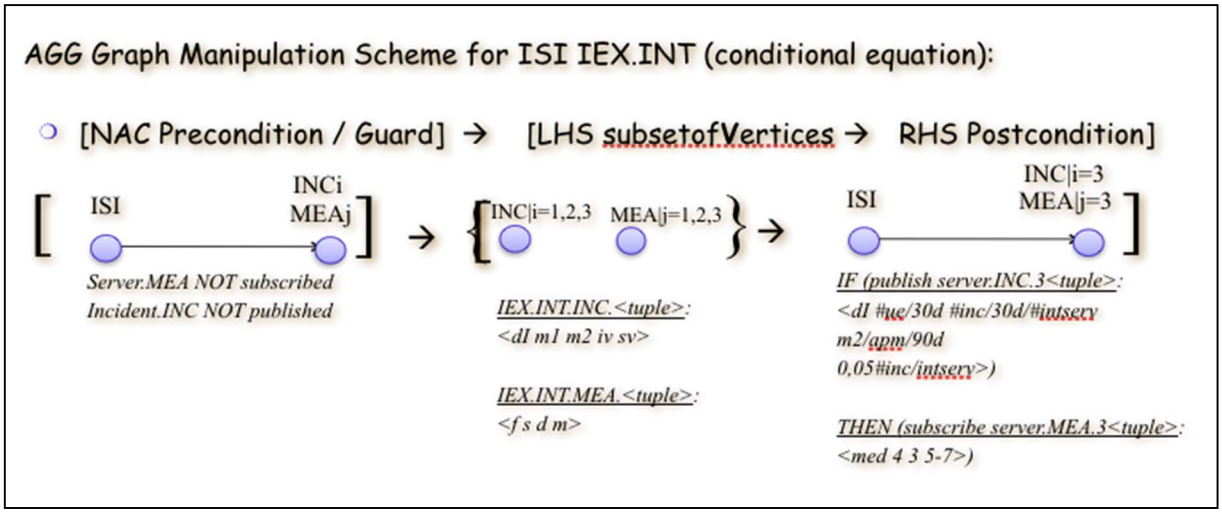

Figure 5: Construction of ISI Type Graph of IEX.INT

Raw data is generated by means of ISI sensor technologies, measurement, supervision equipment applying any kind of logging metering's of incidental events marked by indicator values like 'often', 'few' etc. However in order to encounter raw metering data, the related data<n-tuples> should be published into the ISI Data Lake in the same way as any other item to be communicated. An <n-tuple>, in *CSlang* is defined as an Abstract Data Type, e.g.:

with SORTS: meter value domain tuple-space

OPNS: <formalsort … formalsort> => tuple_space

whereas 'formalsort' is to be replaced by any sort of the declaration part;

thus <meter value domain> defines a <3-tuple>

whose actualized sorts are to be replaced by terms of the corresponding sort,

e.g. <meter:"id" value:"measurement" domain:"name">

In figure 5 a partial graph is given that represents the continuous variables (processes) of generating the security indicator IEX.INT by a conditional equation with the three elements of specification: the guard or precondition, the event head (lefthand side of the equation), and the event tail (righthand side of the equation) with the meaning: when the guard is fulfilled, the equation (with LHS => RHS) is activated, i.e. the undirected edge is turning into a directed edge which means the event is ready to happen. In the case of figure 5 the MEA parameters are finally updated.

Annex A (informative): Proof of Concepts (PoC) - Two Levels of Semantics

A.1 Introduction

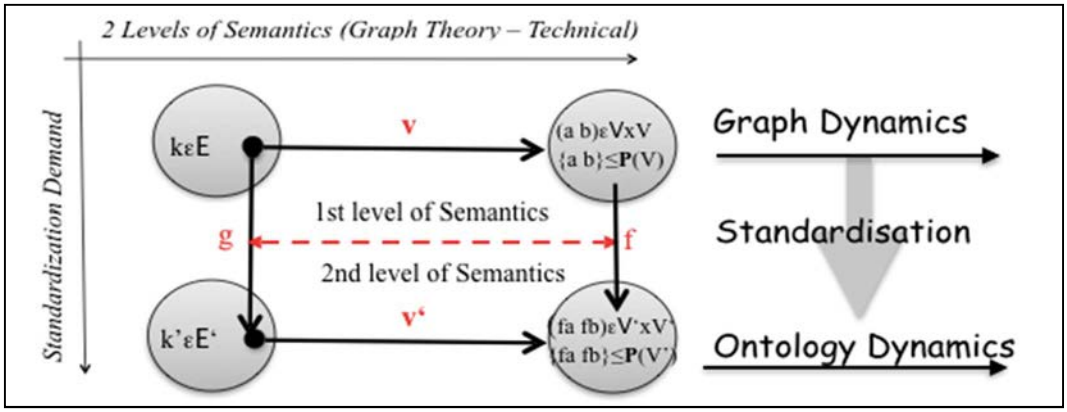

Figure A.1: PoC Schema of 2 Levels of Semantics: Relationship between Ontology and Graph Dynamics

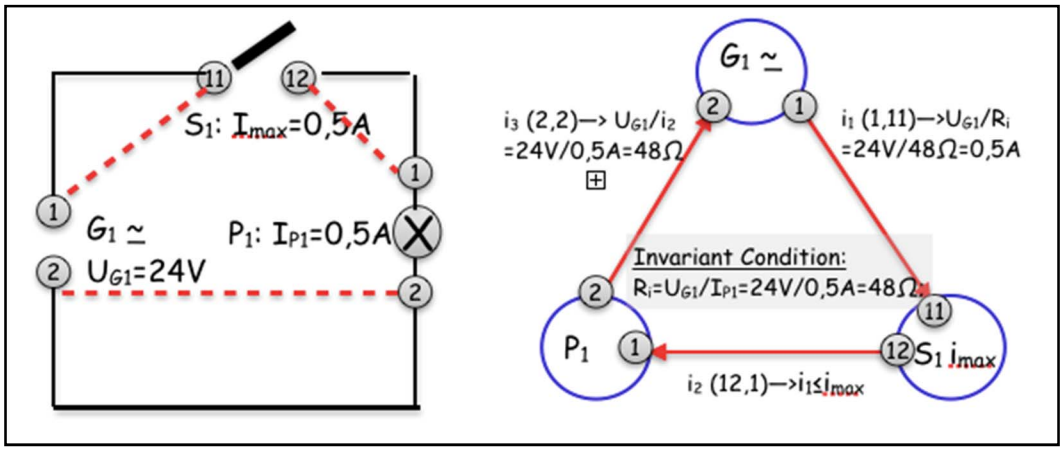

Figure A.2: PoC Schema of 2 levels of Semantics: formal Type Graph (figure right) and descriptive Ontology of a Physical Network (figure left) that is structure-preserving

Annex B (informative): Theories and Formal Methods - Basic Definitions

B.1 Graph Theory

In [i.14] a '*System Design*' is denoted to as an algebra of a set of hierarchically graphs which means a design is a model that fulfils certain rules and constraints, i.e. invariants, stated in a formal language, e.g. a graph manipulation oriented language.

A graph is a triple <E K v> comprising of two sets: set E of Vertices and a set K of Edges, disjunct from E. The third element is called a designer mapping v: K-->P(E) | E x E; whereas every edge kεK is either a mapping onto a subset of vertices vk={a,b} ⊆ E called undirected edges; or onto a pair of vertices: vk=(a,b)εExE called directed edges.

Hence semantics by means of graphs are so-to-say guidelines on how to transform graphs into other graphs by means of tools and by fulfilling formal criteria. The evaluation by simulation comprises the rules of a transformation from graph G: $<E_1\ K_1\ v_1>$ into graph G': $<E_2\ K_2\ v_2>$ and is ascertained by graph theory-based tools.

Hence in the graph model, graph transformations in real systems are related (i.e. homomorph) to process transformations under certain conditions, i.e. the system invariants are not violated or, in other words the processing constraints to proceed are fulfilled (see also Annex A).

B.2 Machine Learning

The example, derived from [i.13], demonstrates a possible coding of characteristics of similar names (see PoCML of clause 4.3): If the input pattern (the query of a name to be checked to be similar to the learned class of names) encodes names as ordered letter pairs, then the name 'H A N S' is coded into the pairs 'ha' 'an' 'ns'. A similar output (target) pattern encodes for every letter of the name both, the position in the name (j=1,2,3,4) and the position in the alphabet (i=1,2,…26), then 'H A N S' is represented by the 4 letters, respectively bit positions (i j):=(8 1), (1 2), (14 3), (25 4).

The *similarity-preserving* mapping:

$$W\ q^p = t^p$$

is the better the more bits are matching between the associative memory W and the query vector q^p; since in the example there are coded letter pairs defining a threshold value, e.g. the 4-letter word 'H A N S' has the threshold value of (4-1). If the query pattern is error prone, e.g. 'H I N S' then the highest threshold value delivers the most similar result. In case the result is not adequate the threshold value should be lowered stepwise to a minimum of 2 which delivers more similar results but which should be reconsidered upon, in order to be adequate or not for the SOC making decisions for an IACS controlled.

This ML algorithm enables SOCs and organizations to make a better decision by analyzing error-prone incident logs in order to identify the appropriate ISI category respectively incident class, e.g. to decide among queries on a set of incidents *being similar* under specific considerations: network DoS, Malicious Code, unauthorized access or usage, propagated incident over multiple components, etc.

Considered from an event graph type point of view (see PoC-GM below), a neuron is like a guess to find a new edge that connects the input pattern q^p to the best target pattern t^p. With respect to the analysis of observed incident patterns it means that the found target pattern is in the class of certain incident patterns the best representant provided the threshold is the highest possible and the learned matrix W is a similar-preserving mapping, i.e. for any decision-taking event p the following mapping holds:

$$t^p = W\ q^p$$

Since an edge respectively an event is a pair of type graph nodes, the actual mapping is rewritten as:

$$(Wq^p\ t^p)$$

with the meaning of the event head performing the computation of the input vector q^p with the learned matrix W that finally results into the target vector t^p of the event tail. Hence the neuron component that performs the computation with the variable W is located in the event head of the enrichment type graph.

B.3 Theory of Data <n-Tuples>

As invented previously during the consideration of the IMA 'data lake', a data<n-tuple> is a finite collection of data elements, as it is defined in [i.21]. Tuples are characterized by both, *the order of elements and their multiplicity*. The elements of a data <n-tuple> are often called 'atoms'. For example, data sets can be represented as <n-tuples> in which both the order and the multiplicity of the elements are disregarded.

In the theory of tuples all elements of an <n-tuple> should be atoms, unlike in lists which may contain sublists. The theory defines the vocabulary of <n-tuples> according to ADT set theory which comprise:

1) the constant <.> denoting to an empty tuple;
2) predicates to find either an atom: atom?(x) or an <n-tuple>: tuple?(x);
3) binary functions: u o x to insert atoms into a given <n-tuple> x.

The *semantics of data<n-tuples>* is defined by a set of *axioms* forall u:atoms and forall x:<tuples>, including properties (notice that the axioms in theory have the same role as an *invariants* in systems, so where is no danger for confusion they can be used them synonymously):

1) generation by insertion;
2) uniqueness of insertion;
3) induction property;
4) decomposition property.

For the management of <n-tuples> of the *ISI data lake* additional relation and functions should be defined that include:

1) a unary predicate denoting to identity of all atoms of the <n-tuple>: same?(x);
2) a ternary predicate denoting to: equal multiplicity?(u x y);
3) the length(<.>) function denotes to number of atoms in the <n-tuple>;
4) the binary count(u x) function denotes to number of occurrences of atom u in tuple x;
5) the binary element$[<.>]_n$ function denotes to the n-th element of the tuple <x>;
6) the ternary altering function alter(x z u) denoting to the tuple x obtained by replacing the z-th element of tuple x by the atom u and, where n is a non-negative number.

Annex C (informative): Authors & Contributors

The following people have contributed to the present document:

Rapporteur:

Jan deMeer, smartspacelab GmbH

Other Contributors:

Herve Debar, Institut Telecom, Vice-Chairman of ISG ISI

Arnaud Fillette, Thales, Secretary of ISG ISI

Gerard Gaudin, G²C, Chairman of ISG ISI

Axel Rennoch, Fraunhofer Institut FOKUS

Philippe Saade, ESI Group

Annex D (informative): Bibliography

- ISO/IEC 27034-3 (03/2016): " Information technology -- Application security -- Part 3: Application security management process".
- MITRE CEET V1.0a: "A Standardized Common Event Expression for Event Interoperability (Common Event Expression Taxonomy)".
- MITRE CAPEC List V1.7.1 (April 2012): "Common Attack Pattern Enumeration and Classification".
- Club R2GS 4-page data sheet V3 (2012): "Presentation of the work in progress" (accessible on ETSI ISG ISI portal).
- Club R2GS presentation V4 (March 2012): "The Club and its objectives" (accessible on ETSI ISG ISI portal).
- Club R2GS reference framework V1.0 (December 2009): "A security event classification model at the heart of SIEM approaches" (accessible on ETSI ISG ISI portal).
- JAVAspaces® Technology Oracle Corporation.

NOTE: Available at https://www.oracle.com/technetwork/articles/java/javaspaces-140665.html.

- TSpace® University of Toronto.

NOTE: Available at https://tspace.library.utoronto.ca/about/.

History

Document history		
V1.1.1	February 2019	Publication