CHAPTER 5

Integrity

"Integrity" is one of the "CIA triad" of key information-security system properties, which are

- Confidentiality
- Integrity
- Availability

People also consider additional properties such as

- Authenticity
- Accountability
- Non-Repudiation
- Reliability

which all are defined in [3]. Notice that information security (infosec) and cyber security (cybersec) are not the same thing. Cybersec involves infosec, but some digital-computational systems not only hold information but engage in behaviour – control systems, for example – and cybersecurity also requires that the behaviour of the system is not subverted. I am concerned here primarily with the integrity of safety-related systems, many of which are control systems of some sort.

5.1 Definitions in Engineering Standards

I elaborate first some definitions of system integrity in relevant engineering standards.

From IEC 62443-1-1 [3]

> *2.1.55*
> *integrity*
>
> *quality of a system reflecting the logical correctness and reliability of the operating system, the logical completeness of the hardware and software implementing the protection mechanisms, and the consistency of the data structures and occurrence of the stored data*
>
> *Note to entry: In a formal security mode, integrity is often interpreted more narrowly to mean protection against unauthorized modification or destruction of information.*

From IEC 61508-4:2010 [1, Part 4, Definitions and abbreviations]

> *3.5.4*
> *safety integrity*
> *probability of an E/E/PE safety-related system satisfactorily performing the specified safety functions under all the stated conditions within a stated period of time*
>
> *NOTE 1 The higher the level of safety integrity, the lower the probability that the safety-related system will fail to carry out the specified safety functions or will fail to adopt a specified state when required.*
> *NOTE 2 There are four levels of safety integrity (see 3.5.8).*
> *NOTE 3 In determining safety integrity, all causes of failures (both random hardware failures and systematic failures) that lead to an unsafe state should be included, for example hardware failures, software induced failures and failures due to electrical interference. Some of these types of failure, in particular random hardware failures, may be quantified using such measures as the average frequency of failure in the dangerous mode of failure or the probability of a safety-related protection system failing to operate on*

demand. However, safety integrity also depends on many factors that cannot be accurately quantified but can only be considered qualitatively.
NOTE 4 Safety integrity comprises hardware safety integrity (see 3.5.7) and systematic safety integrity (see 3.5.6).
NOTE 5 This definition focuses on the reliability of the safety-related systems to perform the safety functions (see IEV 191-12-01 for a definition of reliability).

3.5.5
software safety integrity

part of the safety integrity of a safety-related system relating to systematic failures in a dangerous mode of failure that are attributable to software

3.5.6
systematic safety integrity
part of the safety integrity of a safety-related system relating to systematic failures in a dangerous mode of failure

NOTE Systematic safety integrity cannot usually be quantified (as distinct from hardware safety integrity which usually can).

3.5.7
hardware safety integrity
part of the safety integrity of a safety-related system relating to random hardware failures in a dangerous mode of failure

NOTE The term relates to failures in a dangerous mode, that is, those failures of a safety-related system that would impair its safety integrity. The two parameters that are relevant in this context are the average frequency of dangerous failure and the probability of failure to operate on demand. The former reliability parameter is used when it is necessary to maintain continuous control in order to maintain safety, the latter reliability parameter is used in the context of safety-related protection systems.

The IFIP WG 10.4 definitions [1] take dependability to be the ability to deliver service that can justifiably be trusted, and include integrity amongst the dependability characteristics to mean the absence of improper (i.e., unauthorised) system alterations:

> *integrity: absence of improper system alterations.*

For IEC 62443, integrity is a "quality of a system" – we can take it that "quality" means "property". For IFIP WG 10.4, integrity is a state of the world (system changes since initial configuration are all "proper")[1]. For IEC 61508, integrity is a probability, and thus (according to Markov and since) a number between 0 and 1. These are three different categories of (abstract) object. When we talk about integrity, and mean it to be an important property of a running system, we had surely better make sure we are all talking about the same thing! And these are not the only three options, as we shall see.

5.2 Systems and Their Quirks

Humanly-designed systems are deliberately causal objects. Components are designed to have specific causal effects on the environment in which the system operates, and/or on other system components or parts. (A distinction between components of a system and its parts lies outside the scope of this note, so I shall use the terms "component" and "part" indistinguishably.) Components of a system which engage in behaviour, namely reacting to states of the system and available information by effecting changes in state or information, I call agents. Agents may be human, or they may be inanimate.

It is as well to say what a system is. A system is a collection of agents, that is, a collection of entities which engage in behaviour. A system has a boundary, namely

1 However, consider software updates, for example the automatically downloaded and installed security "patches" which modern all-purpose operating systems such as Windows and Mac OS offer. We can take it that these are all "proper" in the sense of the definition. So "proper" looks as if it is to be a sociotechnical term, say similar in meaning to "appropriately authorised". If so, it follows that IFIP-integrity has a sociotechnical meaning. Now, consider a case in which code is downloaded from the operating system provider which causes an essential function to fail, for example the February 2014 "go to fail" bug included in an Apple operating-system software update [2]. According to my conception (below), the operating system lost functional integrity. But the system was appropriately authorised/"proper". So it retained IFIP-integrity even while losing functional integrity.

the distinction between entities which belong to the system and those which do not, and it has an environment, namely those entities which do not belong to the system but interact with it, that is, engage in relations with system entities and have causal power to modify those relations over time.

Systems sometimes do what we want, and sometimes not what we want. They sometimes fail to do anything when we wish them to do something. Systems may be purely physical, engineered in some way, but they also may include components which are human agents, in which case they are called sociotechnical systems. A railway train is a sociotechnical system, with a physically engineered train on physical track, a remote human controller and a driver responding to physical-train dynamics, signals and other events. A signalling system nowadays has many more purely physical components than it did when people in signal boxes moved mechanical components activating semaphore signals. An important part of a modern signalling system and its operation is the display of information to a train controller, hisher processing of that information and the decisions and signalling actions which result. Information, its veridicality and its flow, is an important component of many sociophysical systems such as train operation. Train operation itself is part of a more complex sociotechnical system, namely railway operation.

Engineered systems are usually teleological, that is, they were designed by people with a specific function or a specific goal in mind. Some software-system experts such as Michael Jackson [6, 7] suggest that system requirements are best conceived through imagining and then writing down – specifying – a way you wish the world to be (and which it currently is not, or more generally is not certain to be for a desired length of time), without mentioning a specific mechanism by which this is to be achieved (the system to be defined). The system specification then represents the design of an object which achieves that desired effect. The system works if we can show that the design specification fulfils the requirements specification. Such an approach makes clear the distinction between requirements specification and design specification, and helps establish the general malfunction taxomony below, but will otherwise play no further role here.

Not only teleological systems as a whole, but their agents, have a more or less specified function. Agents will typically have other behaviour which is not part of the system function. Circuit boards bathed in acid engage in different behaviour that may well not be part of their specified function. Such boards will not be bathed in acid while executing their system function (the enviromental conditions under which they operate as intended are usually specified explicitly). Human agents eat and sleep, watch movies, and engage in other behaviour not relevant for system causal operation. We can call the behaviour whose execution is causally critical for correct system function as system-functional behaviour, or functional behaviour for short. It is good practice explicitly to specify the functional behaviour of agents, but such specification is often incomplete.

Systems may malfunction in many different ways. Here is a crude taxonomy. Malfunction may be inadvertent, or may be deliberately induced.

- A system may encounter an environmental situation for which it has not been conceived or designed, and behave in an inappropriate way. I call this a requirements error. The system requirements did not cover all situations to be encountered – they were incomplete.
- A system may have a flaw in its design or implementation, so that it reacts to a foreseen environmental situation, something covered by the requirements specification, in an inappropriate way. This is a design or implementation error.
- Specific agents in a system may malfunction. That is, in circumstances in which they previously behaved appropriately, they no longer do so. A circuit board may "burn out", classified by IEC 61508 as a random (hardware) failure. A human agent may fail to register and act on crucial information. A human-agent malfunction is often said to be a human error.
- Systems are not static in the sense that components change in ways other than as part of their function. Components require maintenance, physical attention paid over time to keep them playing their designed functional role. Both physical and human agents must often be interchangeable – circuit boards can be swapped out for newer boards; human operators go "off shift" and are replaced by other operators. We can call all these phenomena functional maintenance. During functional maintenance activity, it may be that certain system components lose or change part of their functional behaviour. I will say that the agent loses functional integrity.
- Other changes may occur to systems other than through functional maintenance. Some human may deliberately try to compromise functional integrity by introducing components, or changes to components, with a different functional behaviour than expected or specified. Such components, or portions of

components, are often called malware, which in our use here may be hardware or software or both. When malware is introduced into a system component, it loses functional integrity.

I have said above what it is to lose functional integrity, but not what functional integrity is. This may be grasped by saying what it is not to lose it. I propose the following characterisation.

- functional integrity is the property of a system or component that its system-relevant behaviour remains the same.

It remains of course to say what "system-relevant behaviour" is.

- system-relevant behaviour is behaviour of a system or a component of a system which contributes causally to the fulfilment of some part of the system requirements specification

Malware in a purely software-driven system may cause it to behave in ways nonconformant with its specification or legitimate expectations of its stakeholders. However, malware in a sociotechnical system does not always affect system functionality in quite the same way. Consider the following situation. A physical system may have as (partial) function to provide information to a human agent, who then acts upon that information. Malware may corrupt the information provided, so that a picture of the world (a partial world-state) is displayed which is at variance with the real partial world-state. This may happen even though the system retains functional integrity as explained above. (This situation occurs also in sociotechnical systems which do not display malware influence [10].) A human agent, an operator, may take that non-veridical information as veridical and act according to it, thus propagating behaviour appropriate to a situation, which is not the real situation, through the system. Or the operator may notice an anomaly and take action to validate the information or otherwise mitigate its effect upon system behaviour, thereby "smoothing" the effect of the anomaly. Human agents are traditionally used in critical system operations to provide such an anomaly-smoothing role. But also, as is well-known, they indulge in sui generis inappropriate action, even on veridical information, for a variety of reasons which fall into the category of human error. Analysing the conditions under which a veridical/anomalous causal chain of information is passed through a human operator I have called “semantic safety”[1] [11]. Analysing semantic safety requires assigning an explicit meaning to, say, information displayed to an operator, which meaning, $Meaning(D)$, is defined by the physical aspects of the display D. One can then ask if and how the operator uses $Meaning(D)$ in hisher further deliberations and actions within the system operation. For example,

Perceive: the operator has display D within hisher sensory field

Attend: the operator assimilates $Meaning(D)$

Reason: the operator deliberates, given $Meaning(D)$ and the procedures, on the necessity or appropriateness of an action A

Decide: the operator decides to perform A;

Intend the operator forms the intent to perform A subsequent to hisher decision[2];

Act: the operator executes the action A

I call this the PARDIA information-processing decomposition of operator actions and we have successfully used it in analysing crew actions in commercial transport aircraft accidents [9, 10]. It appears to be similar, although more finely-grained, than the OODA classification [15]. Such sequences are often called information-processing models in engineering psychology [14]. For each step of PARDIA, there are examples in aviation in which the human information-processing sequence failed at that step.

In a situation in which $Meaning(D)$ is non-veridical (for whatever reason) and the operator O induces actions which are inappropriate, the functional integrity of the system causally downstream from O may remain intact. However, because of the actions taken on misleading critical information in $Meaning(D)$, the computation causally downstream of O has been corrupted. I wish to include this situation also as a loss of integrity, but it is not due to a lack of functional integrity causally downstream of O. The situation causally downstream from O has been generated by the non-veridical information in $Meaning(D)$. So it seems that we need a notion of information integrity.

1 The original purpose of this notion was to trace causality through display systems, in order to be able apply the concepts of the functional safety standard IEC 61508 also to critical operator-informational systems in safety-critical applications, as in air traffic control, or nuclear power plants.

2 In most cases, decision and intention may be conflated. But there are some situations in which an operator can decide, but be interpreted as failing to form an intent to act on the decision; a pilot awaking from G-induced loss of consciousness, for example. This topic is outside the scope here.

A system includes generally two types of information:

- information which reflects real-world parameters. This information might be veridical (has the same value as the real-world parameter) or non-veridical (has a different value from that pertaining in the real world).
- information internal to the system, which has limited or no correspondence with "outside" parameters

We deal with these two types of information in separate clauses of the definition of information integrity. We may characterise information integrity as follows.

- information integrity is the property that the meaning of the information held at any state St of the system Sys is conformant with
  - either the real world (that is, the information corresponding to real-world parameters is veridical) or
  - veridical information held at other states St_1 of the system, transformed by the functionally-correct transformations applied by Sys to St_1 which result in St.

The situation is described in the Causal Control Flow Diagram (CCFD) in Figure 7.1. A CCFD is a mathematical discrete directed graph, with nodes (boxes) and arrows. The arrows indicate causal relations between the nodes: the node at the tail of an arrow is a necessary causal factor of the node at the head (with the exception of "OR" nodes, which are purely formal, and which play an intuitively obvious semantic role, in that one or other of the factors at the tail are a necessary causal factor of the node after the "OR" node). A technical test, the Counterfactual Test [7, 8], establishes whether a node is a necessary causal factor of another.

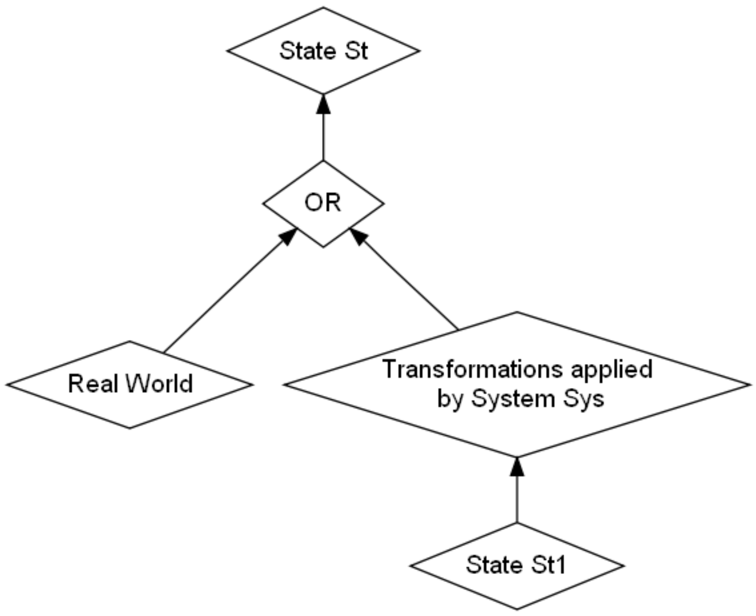

Figure 5.1: Information Integrity Causal Control Flow Diagram

5.3 Summary

The IFIP WG 10.4 definitions take dependability to be *the ability to deliver service that can justifiably be trusted*[1], and include integrity amongst the dependability characteristics to mean the absence of improper (i.e., unauthorised) system alterations [1].

IEC 61508-4:2010 defines safety integrity (Clause 3.5.4) to be the probability of an E/E/PE safety-related system satisfactorily performing the specified safety functions under all the stated conditions within a stated period of time [2, Part 4, Definitions and abbreviations]

1 In the original vocabulary published in [13], this was elaborated as *[t]rustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers*, which makes clear its intended sociotechnical nature. In the later vocabulary in [1], in which the shorter form here quoted appears, this explicit sociotechnical formulation is modified. The notion of "dependability" is not our topic here.

As noted in Section 5.1, for IFIP, integrity is arguably a sociotechnical state of the world (system changes since initial configuration are all "proper"). For IEC 61508 integrity is a number between 0 and 1, a probability. For IEC 62443, it is a system property. These are very different categories of (abstract) object.

None of these definitions seems quite right for our purposes. The IFIP definition arguably makes the notion of integrity dependent on the notion of authorisation, that is, on a human qualification. But what if this qualification is disavantageous? Say, an authorised maintainer inserts malware? The person was authorised, performed hisher actions as required, but the object heher was manipulating, the replacement code, was compromised – it was not identical with the replacement code which others in the authorisation chain had intended. It may be considered that the user might have been authorised, but the replacement code also needs to be authorised in some fashion. The question then arises what authorisation structure for code is necessary and/or sufficient. The further question arises: necessary and sufficient for what? We could reply: for the functionally-correct behaviour of the system; then we are close to the definitions I have proposed here.

Integrity of function as defined here is not covered by the IFIP definition, either explicitly or implicitly. Code can be modified through authentic modifications inserted by an authorised maintainer, but the newly modified code could fail to perform a key function which was correctly performed by the original code, as for example in [2]. The code may have retained its IFIP integrity, but it has not retained its functional integrity according to my definition – it performs certain functions differently (and incorrectly) from before.

A characterisation of integrity as a probability, as in IEC 61508-4:2010, does not seem to fit at all with common use of the word. What is it to retain integrity, when integrity is a number? Suppose the code is changed, authorised or unauthorised, and safety functions are executed to different reliabilities than before, but somehow, under a specific set of conditions and defined time period, the probability that the safety functions will be executed remains the same? Many would say the integrity has been compromised (certain critical functions do not work when, or work to a different likelihood than, they did before) but IEC 61508 requires us to say it has remained the same.

I suggest that the notions of integrity of function and integrity of information, as I have defined them, are two key facets of integrity. Here, I consider information not to be the same as data, but to be interpreted data, that is, data-with-semantics[1]. If the semantics changes but bits remain the same, say bits which encoded numbers as floating-point are interpreted instead as fixed-point numbers, then information has changed even though the data have not. If the encoding of, say, characters in bits changes, say, from ASCII to Unicode, then the data have changed but the information not. It is the information in an operator display, not the bits – say, the arrangement of pixels – which contributes to semantic safety.

1 It is common to say that data is the representation of information inside a system. I do not necessarily endorse this view, but I do sympathise with it.

Bibliography

[1] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell and Carl Landwehr, Basic Concepts and Taxonomy of Dependable and Secure Computing, IEEE Trans. Depend. Sec. Comp. 1(1):1-23, 2004. Available from https://www.nasa.gov/pdf/636745main_day_3-algirdas_avizienis.pdf, accessed 2017-12-01.

[2] Paul Ducklin, *Anatomy of a "goto fail " – Apple's SSL bug explained, plus an unofficial patch for OS X!*, blog post at https://nakedsecurity.sophos.com/2014/02/24/anatomy-of-a-goto-fail-apples-ssl-bug-explained-plus-an-unofficial-patch/, 2014-02-24. Accessed 2017-12-01.

[3] International Electrotechnical Commission, IEC 27000:2016 Information technology – Security techniques – Information security management systems – Overview and vocabulary, 2016.

[4] International Electrotechnical Commission, IEC 61508:2010, Functional safety of electric/electronic/programmable electronic safety-related systems. 7 parts, IEC, 2010.

[5] International Electrotechnical Commission, IEC TS 62443-1-1:2009, *Industrial communication networks - Network and system security - Part 1-1: Terminology, concepts and models*, 2009

[6] Michael Jackson, *Software Requirements and Specifications*, ACM Press/Addison-Wesley 1995.

[7] Michael Jackson, *Problem Frames*, ACM Press/Pearson Education, 2001.

[8] Peter Bernard Ladkin, Causal System Analysis, e-book, RVS-BI, 2001. Available from https://rvs-bi.de/publications/books/CausalSystemAnalysis/, accessed 2017-12-01.

[9] Peter Bernard Ladkin, *The PARDIA Classification*, Chapter 18 of [8].

[10] Peter Bernard Ladkin, *Verbal Communication Protocols in Safety-Critical System Operations*, in Dafydd Gibbon and Alexander Mehler, eds., Handbook of Technical Communication, Mouton de Gruyter, Berlin, 2012.

[11] Peter Bernard Ladkin, *Message to the System Safety Mailing List*, 2016-08-25. Available at http://www.systemsafetylist.org/2797.htm, accessed 2017-12-01.

[12] Peter Bernard Ladkin, *Digital System Safety*, textbook, to appear, 2017-18.

[13] Jean-Claude Laprie (ed.), *Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese*, Dependable Computing and Fault Tolerance Vol. 5, Springer-Verlag, 1992.

[14] Skybrary, *Information Processing*, no date. Available at https://www.skybrary.aero/index.php/Information_Processing, accessed 2017-12-01.

[15] Wikipedia, *The OODA Loop*, no date. Available at https://en.wikipedia.org/wiki/OODA_loop, accessed 2017-12-01.