ARTICLE:

Assigning IACS cybersecurity responsibility conformant with the UK Network and Information Systems Regulations 2018

By Peter Bernard Ladkin and Martyn Thomas

Introduction

Computers incorporating digital electronics based on operational logic called 'software' used to be found in climate-controlled specialist environments, with dust-filtering ventilation and underfloor cooling circuitry, occupying hundreds of square metres of floor space. Their 'input' was provided on packs of special Hollerith-encoded cards, and the output was on continuous reams of concertina-foldable paper. While the telephone service was used to connect such machines, it was not otherwise dependent on them, any more than much of the rest of daily life. Now, telephone switching is accomplished by the descendants of such computers, as is much switching logic in engineering, and indeed the telephones themselves, if mobile, are such computers, which even though tiny are far more computationally powerful than their forebears. Digital computers have become embedded in engineering applications of all sorts, from automobiles to telephones to broadcast media to industrial process-plant control. They have become ubiquitous in what is called 'critical infrastructure', the engineering developments which provide the environment for daily life as we now know it.

Along with this pervasiveness with its myriad advantages have also come the weaknesses of digital-computer-based engineering, notably the relative lack of dependability of software-based devices compared with other engineered non-digital artefacts. The characteristics of software and its quality are not generally known amongst non-specialists. We have addressed software quality elsewhere, e.g., (Ladkin, Littlewood et al 2020, Ladkin and Thomas 2020). This paper depends on those observations.

We are concerned in this paper primarily with the cybersecurity vulnerabilities of industrial automation and control systems (IACS, or ICS). Two of the reasons for such a restriction are that

> (i) there is a specific multilateral organisational structure involved, namely system integrator (SI), original equipment manufacturer (OEM), and operator (OP) (see below), and incident reporting and response requirements require some negotiation between these entities, whose organisational and legal interests may not always cohere;
>
> (ii) some of the companies involved in IACS have already made suggestions in public as to how responsibility is in their view to be assigned.

A mandate-responsibility model such as we develop here is tailored specifically to the given organisational structure (SI, OEM, OP plus the NIS-mandated organisations Competent Authority (CA), Single Point of Contact (SPOC), and Computer Security Incident Response Team (CSIRT)). We would expect that other critical-infrastructure areas have similar constraints, namely organisational structure and the views of relevant interested parties, and therefore a mandate-responsibility breakdown for these is possible, but not necessarily identical to the one we derive for IACS here.

Industrial process plants incorporating IACS are designed and assembled by an organisation we can call the system integrator (SI), using components from suppliers, which we can call original equipment manufacturers (OEMs) and are operated by an organisation we call simply the operator (OP). For example, a power station might be designed and built by one organisation, say Drax by the Central Electricity Generating Board (CEGB); involve equipment from an OEM, say Siemens S-series or Schneider Modicon controllers, to name two of the most widely used, and be operated by a third, Drax Power Limited (in this example, SI responsibilities would inhere in the legal successor of the CEGB, which no longer exists). When something is not right, or goes wrong, there are a number of organisations to which to look for remedy, for instance:

> Perhaps the design was inadequate for a cyber-insecure world (for example, there is no duplicated functionality so that when one digital control channel is compromised, there are no parallel second and third channels to take over control).
>
> Maybe specific equipment exhibits exploitable vulnerability (for example, an exploit is known for a specific controller produced by an OEM).
>
> Possible vulnerabilities lie in operations (for example, the operator has not yet performed software updates to address a vulnerability identified by an OEM for which patches, software updates, are available).
>
> It could be that the operator has not sufficiently 'firewalled' its critical systems from digital communication paths to the 'outside'.

The mandate-responsibility model uses the notions of *mandate*, *duty* and *duty holder*. We identify first a *mandate*, then attempt to derive *duties* and *duty holders* from this. There are two steps. First, a mandate is formulated. A *mandate* is a general requirement, an obligation, on as-yet-unspecified agents, a form of saying 'the world should be so' without saying how it gets to be so or who makes it so. Mandates are derived by considering the nature of software and its failures, errors, and loss of functional integrity, and what is necessary to minimise the consequences and to remedy the situation. Second, to fulfil mandates there must be actors undertaking performances which lead to that fulfilment. Such performances we call *duties*, and such actors *duty holders*. Duties and duty holders are identified by considering the organisational infrastructure and relations between stakeholders more closely.

For example, if it is intended that a CSIRT shall act as a repository for reports and analyses of significant cybersecurity incidents, then the following must occur:

> given that an incident has occurred, it must be observed
>
> …. and recorded
>
> …. and the record/report communicated to CSIRT
>
> …. and the incident deemed to be significant.

All of that constitutes a mandate. Given the mandate, the questions arise:

> who observes the incident?
>
> who shall report it (and how)?
>
> who shall communicate it to CSIRT (or take responsibility that it is so communicated)?
>
> who is the deemer of significance?

Sometimes the answer is evident:

(i) an operator observes an operating anomaly; a specialist is tasked by the operator to investigate, and determines a cybersecurity incident has occurred; the operator reports it to CSIRT;

(ii) an OEM discovers a vulnerability in a device it supplies to many plants; the OEM analyses the vulnerability, devises a patch, and reports it to CSIRT.

But sometimes answers are not evident:

(i) If CSIRT is informed of an incident involving amongst other things an OEM's equipment, how does CSIRT behave vis-a-vis the OEM? When CSIRT merely informs the OEM, is there any obligation for the OEM to respond (say, with a patch, or with a recommendation for the operator)?

(ii) If an OEM becomes aware of an incident involving its equipment, does the OEM inform CSIRT; or does the OEM first analyse the incident for significance, and inform CSIRT only if it deems the incident significant?

A detailed regulation may well determine who shall act and proffer answers to such questions, for example:

(i) an operator observing an anomaly shall report to CSIRT, CSIRT requires certain further details from the operator, CSIRT then analyses for significance and deems significant or not; alternatively, an operator analyses an anomaly for significance, and refers to CSIRT using a standard report format if the anomaly is deemed significant;

(ii) an OEM informs CSIRT of vulnerabilities known to it in its equipment; CSIRT deems these significant (or not); alternatively, the OEM analyses for significance, and informs CSIRT only of significant vulnerabilities.

Many people associate cybersecurity vulnerability in an industrial plant with shutdown[1] or impedance of operations. But unavailability is not the only concern. Most industrial plants conduct operations which have safety considerations; when precautions (so-called 'safety functions') are not taken then the risk of damage to people, valuable equipment, or the environment would be too high. Functional safety standards require safety functions and directly address the possibility of a loss of functional integrity[2] in the implementation of a safety function (IEC 61508). As digital components become ubiquitous, many safety functions become partly or wholly implemented through digital equipment. Vulnerability to cyberattack can enable the functional integrity of the safety function to be compromised, and thereby the continued operation of the plant to become unacceptably risky.

Some aspects of cybersecurity are addressed in the applicable functional safety standards, but many argue (and have commented to the International Electrotechnical Commission (IEC)) that more, and more specific, measures should be required (Ladkin 2017, Chapter 13: Cybersecurity in IEC 61508:2010 and IEC 61511:2016).

Mandates and responsibilities in the implementation and operation of IACS

Many consumer products come from a single organisation, a company, which incurs various legal duties to those who purchase and use the product, for example that the product does what it purports to do at purchase, and that it operates in a suitably safe manner (EC 765/2008).

[1] Here, an orderly, intentional cessation of operations.

[2] Integrity is one of the 'CIA triad' addressed by much cybersecurity literature (Ladkin 2017, Chapter 8: The CIA Triad). It is a concept which does not have a uniform definition in the engineering literature, but this potential weakness can be effectively addressed (Ladkin 2017, Chapter 5: Integrity; Chapter 7: An Example in which System Integrity is Critical).

An IACS is part of a much larger system, which has many components. Such components are provided often by different specialist *manufacturers* (which we have called OEMs), and they are designed into and integrated into one (hopefully) coherent, large, complex system by an engineering organisation, often separate from the OEMs, which we have designated the *systems integrator* (SI). The system is operated by an *operator* (OP), who is usually different from the OEMs and the SI. The system in operation will be maintained (that is, rendered continuously operable) by a combination of these entities, or maybe by other organisations.[3]

We have indicated that one main concern is plant safety. We compare the situation briefly with that in commercial air transport safety. The operator of an aircraft is an airline. The aircraft manufacturer is equivalent to the systems integrator of an industrial plant, putting components such as engines, avionics, seats and toilets supplied by OEMs together into one airplane. The manufacturer of an aircraft, though, is often also responsible for the aerodynamic engineering and the specific construction of the body, wings and tailplane of the aircraft, whereas a similar manufacturing role may well not be played by the SI of an industrial plant.

For international commercial aviation, the duty of safe transportation is carried by the airline. This is the result of a series of international conventions, starting in 1929 in Warsaw, now under the auspices of the United Nations. Even when physical parts of an aircraft fail catastrophically, the convention holds that it is the airline which is responsible, and must recompense victims and their families. (Such unambiguous liability is then usually negotiated further. If the accident was caused for example by component failure, the airline's insurers will typically approach the component manufacturer for compensation.)

For industrial plant, there is no such international convention, because industrial plants have one fixed location. However, component manufacturers and systems integrators operate internationally and it can be to everyone's advantage to encounter formal similarities, even identity, between countries' rules governing duty and duty holders in safety aspects, and, increasingly, in cybersecurity aspects which may affect safety.

We seek a solution to the issue of IACS system quality and responsibility for it, based on engineering considerations. The aim in such engineering is not perfection, as it would be in science or mathematics; the aim is the fitness of a system for its purpose. The objective is to make risks tolerable (or, in English law, 'as low as reasonably practicable' (ALARP) (citing Asquith LJ)[4]), but not necessarily zero (Health and Safety Executive 2001).

One industry view

Cybersecurity is a particularly difficult issue with safety-critical systems. Traditional concerns about safety assume that the combination of unfortunate factors which lead to damage is inanimate, occurring more or less by chance, which chance will generally be low if the system has been specified, designed and built according to best practice. If people are actively attempting to contravene the integrity of a system, the occurrence of a 'combination of unfortunate factors' is no longer a matter of chance; if someone knows about them and there is a way to bring them about, then whether they occur or not is a matter of human intent, no longer of happenstance (Ladkin 2017, Chapter 14: The Notion of Security Risk).

[3] Engineering standards for such systems often also consider end-of-system-life issues, system decommissioning. We do not consider system decommissioning aspects here.

[4] *Edwards v National Coal Board* [1949] 1 KB 704, [1949] 1 All ER 743, 65 TLR 430, [1949] 3 WLUK 82, (1949) 93 SJ 337, [1947-51] CLY 6274, Asquith, LJ at 712; but see legal practitioner texts on subsequent case law and comments and observations in relation to this case by superior courts, including, but not limited to Professor Andrew Tettenborn, general editor, *Clerk & Lindsell on Torts* (23rd end, Sweet & Maxwell, 2021) Consolidated Mainwork Incorporating First Supplement, Chapter 12 – Employers' Liability, Section 3 – Breach of Statutory Duty, paras 12-46 – 12.47.

One major international component manufacturer sees the duties and duty holders concerning cybersecurity as follows (here, OEMs are referred to as 'manufacturers') (Siemens 2017) (bullet points in the original):

> '... there is a need to clearly delimit the responsibilities related to security among the economic players involved (operators, system integrators, and manufacturers) according to their specific roles:
>
> - The operator or asset owner is responsible to specify the security requirements for the solution in such a way, that the solution fits to the risks and intended use/operational environment. This means that the operator/owner is, in the first place, responsible for ensuring that the security is kept up-to-date within the life-time of the solution. However, these activities may be supported by the integrator, service provider, or manufacturer.
> - The responsibility of the integrator is limited to the compliance with the security specifications of the operator and the secure deployment of the system or solution, but does not in principle cover operation or maintenance (including regular security updates).
> - The responsibility of manufacturers covers the security features of the products they market, and can only be related to the 'intended or reasonably foreseeable use' of a product, as determined in particular in the user instructions. Security-related declarations or certifications only apply to the referred product or solution, but are no longer valid in case of major adaptations/changes of the product or solution, or in the case of their usage outside or beyond the documented 'intended use'.'

This suggests that, in Siemens' view, translated into our preferred terminology of duties and duty holders:

> (i) There shall be a set of security requirements for the plant. There is a duty; namely that these requirements 'fit' the intended operation as well as the actual operation of the plant, and that they consider and respond to the identified risks in and around such operation. The duty holder is the operator.
>
> (ii) There is a duty that the plant as built conforms with the specified security requirements. The duty holder is the system integrator.
>
> (iii) The products provided by the OEMs and put together by the system integrator into the deployed system shall have a set of 'user instructions' relating to 'intended or reasonably foreseeable use'. There is also a set of 'security declarations'. These claim cybersecurity properties of the product used in conformance with the user instructions in a 'reasonably foreseeable' manner. The duty is to determine such user instructions and security declarations. The duty holder is the OEM. It is not clear what duty is attached to determining 'intended or reasonably foreseeable use', nor who the holder of such a duty would be.
>
> (iv) There is no definition of duties or their holder(s) pertaining to major changes to components.

We understand that this suggestion is in line with the traditional ways in which responsibility for safety-related matters is distributed in industrial plant. However, we suggest below that questions are raised by the particular nature of software-based systems which are not easily addressed through this framework as it stands.

Mandates and duties

We are concerned with the process of dealing with defects in software-based components of an IACS and include cybersecurity vulnerabilities as defects. Some cybersecurity vulnerabilities arise because of certain weaknesses in standard computer and networking protocols which are known. The protocols do not

necessarily change when weaknesses are known, often because they are pervasive and a large installed base of engineering artifacts depends on their constancy. However, one can consider a general social obligation to mitigate the effects of known vulnerabilities, to work around them or to enhance the operation of the protocol with supplementary measures, in specific cases in which it is necessary. Such an obligation can be formulated, without yet necessarily considering which organisation or person is obliged to carry all or part of it through, or why. This obligation we have called a mandate.

There will be the further question of which organisations shall hold which duties in order to fulfil the mandate. In many cases, duty and duty holder will be obvious. But sometimes it is not so clear how duties should be formulated, even when the mandate itself is clear. In these cases, the separation into two steps, one deriving the mandate, and the second deriving duties and their holders from the mandate along with additional social-structural conditions, becomes helpful.

We have seen that critical software in its current means of production is subject to defects. It can therefore be expected by all parties that software-based components contain design defects (namely defects in the software). We propose first a mandate that

> (M1) Software defects in components of safety-critical systems are made known to relevant parties as soon as reasonably practicable after discovery.

The argument for this mandate is straightforward. A defect might well affect, even invalidate, the assessed argument for safe operation of an industrial plant accepted by the regulator (which we call here the 'safety case' for short). If a defect is identified by someone, but for example the OEM or system operator remain unaware of it, then not only may the plant not be operating with an acceptable degree of safety (if the defect invalidates the safety case), but some antagonistic third party might learn of it, might be able to exploit it, and thereby cause intentional damage.

The procedure of translating the mandate into processes by means of which the relevant parties become aware of a defect is the transition from mandate to a system of duties and duty holders. In the case of knowledge of defects, these processes and ensuing duties and duty holders is largely addressed for some areas of industry through the European Union NIS directive (EU 2016/1148) and we shall have little more to say about it here.

This illustrates the usefulness of our two-step mandate-to-duty approach, in that the mandate (M1) has been clear for at least two decades, since the mid-1990's, and it has thus taken about twenty years to formulate more specific mechanisms, along with duties and duty holders, as embodied in the EU NIS directive in 2016 and its national implementations.

Defects in software-based components

Engineered-system components possess features which limit the environments in which they can be expected satisfactorily to perform their function. Mechanical and electromechanical components often come with a 'data sheet' which expresses such limitations and gives information on the component from which such limitations obviously derive. For software-based components, we call this the 'user manual'. (This is referred to as 'user instructions' in our quotation from (Siemens 2017), for example.) There is an obvious mandate that:

> (M2) the user manual contains all applicable limitations.

(This immediately results in a duty to write down such limitations, and an obvious duty holder, namely the author of the user manual, which is the OEM. We defer this discussion.) Formulation of this mandate leads us to an evident possibility, which is that a defect might be known, without this knowledge having resulted in a correction (modification of the component) or limitation (restriction of the environment to those in which the defect does not manifest deleteriously). A further mandate is needed:

(M3) software defects, when known, result in

correction, or

an explicit limitation in the user manual, or

both (in case of a correction which has consequences for system operation).

(Again, this comes with some easily-identifiable duties, considered below.) We can classify defects into two classes:

(i) Defects which are known at point of delivery of the component (from OEM to SI).

(ii) Defects which are unknown at point of delivery.

Defects known at point of delivery

It may seem odd that if a defect is known to an OEM at the point of delivery, it would not have been corrected. In fact, defect correction in software logic ('bug fixing') suffers from the phenomenon that correcting a defect may itself introduce further defects, which are at that point unknown. Recurrent bug-fixing of this sort may result – and has sometimes resulted – in a system which is less dependable than the original. It is preferable in some cases to leave the code as is, note the defect and its manifestations, and provide users with guidance to avoid manifestation of the deleterious effects of the defect when the code runs. In the informal language of programmers, the defect becomes a 'feature, not a bug', and sufficient guidance in avoiding it shall be included in the user manual according to (M3). This is closely similar to what is done with mechanical and electromechanical equipment – defining operational limitations and placing these limitations in the operations manual for the device.

Given that specific organisations are involved in correction of software logic and/or authorship of the user manual, it is straightforward here to identify duties and their holders. There is:

(D1) a duty, carried by the OEM, to correctly identify operational limitations imposed by the presence of the defect,

(D2) a duty, carried by the OEM, to present those (adequate) operational limitations fully in the user manual,

(D3) a duty, carried by the system integrator, to incorporate those operational limitations in a deployed system, and

(D4) a duty, carried by the operator, to abide by those operational limitations.

It could be argued that these duties follow from the Siemens conception noted above. We know, though, of no explicit argument to that effect. We thus consider it appropriate to set them out as we have done here.

Defects unknown at point of delivery

The issues around defects unidentified at the point of delivery is more complex. If limitations appropriate to those defects should happen to be included in the user manual, then that would be purely through happenstance. We consider the situation in which there are no such limitations. The system integrator includes the device with the defect in the system that is installed. The operator operates the system without consideration of the presence of the defect, since no one knows of it. There are two (sub)cases to consider:

(i) The defect is identified, perhaps by a third party (cybersecurity firms do so regularly).

(ii) The defect remains unidentified.

When the defect is identified, (M1) says it shall become known to relevant parties. There are now in the US and throughout the European Union organisational structures and procedures by means of which knowledge of the defect and eliminations/mitigations of its consequences are distributed, namely through US ICS-CERT and through the competent authorities/CSIRTs specified in the European NIS directive[5] (EU 2016/1148). (We use the term CSIRT henceforth to include both CERTs and CSIRTs.) We now consider what duties consistent with the existence of a (properly operating) CSIRT-based infrastructure might exist.

Duties following a CSIRT procedural environment

It has become standard in industry that the (benign) third-party discoverer of a cybersecurity vulnerability informs OEM and CSIRT of the vulnerability, and time is given for the OEM to develop an appropriate engineering response (elimination and/or operating limitations and/or mitigation of consequences). When this response is available in full, the discoverer and CSIRT make the vulnerability public (it, and the response, are for example recorded in public databases of vulnerabilities). This way of proceeding has evolved over two decades and many consider it appropriate to consider its incorporation into the legal system, as in the NIS Directive. We note again that not all software defects lead to vulnerabilities, and not all vulnerabilities are the result of software defects.

Given this procedural context and that it is quasi-universally deemed to be an appropriate response to mandates (M1) – (M3), we can discuss duties that may desirably follow in this context. When a vulnerability is discovered by a third party, according to (M1) it shall be made known to the relevant parties. Since the third party is the (presumed) sole possessor of that knowledge, a duty to communicate, or to enable communication of, this knowledge to relevant parties follows in order for the mandate to be fulfilled. When a CSIRT-based infrastructure exists, conformant with industry best practice, this duty can be made more concrete as follows:

> When a vulnerability is discovered by a third party, according to (M1) and the existing infrastructure that third party holds:
>
> > (D5) a duty to inform CSIRT in as full detail as possible of its knowledge of the vulnerability;
> >
> > (D6) a duty to inform the OEM similarly;
> >
> > (D7) a duty to maintain confidentiality between itself, CSIRT and OEM about the vulnerability until such time as CSIRT decides to make the knowledge and engineering response public.
>
> According to (M2) and (M3) and the existing infrastructure, the OEM holds:
>
> > (D8) a duty to acknowledge and characterise the vulnerability fully;
> >
> > (D9) a duty to analyse the vulnerability and develop countermeasures (elimination and/or mitigation and/or operating limitations);
> >
> > (D10) a duty to convey knowledge of the vulnerability and countermeasures to users of the equipment as a matter of urgency.

A subsidiary duty arises from (D10). In order to facilitate knowledge of the vulnerability and countermeasure to users, the OEM holds thereby:

[5] NIS is somewhat limited, in our view, in that it only applies to systems providing a critical service to a large number of people/customers. This leads to anomalies, such as the digital communications/networking infrastructure of nuclear power plants (NPPs) in Germany not falling technically under the NIS, because of the limited number of providers/operators of NPPs (Böllmann 2018). A similar phenomenon occurs with the communications infrastructure of air traffic management and control (de Haan, Ladkin and Sieker 2016), an issue being addressed in a Eurocontrol Task Force on the Resilience of Digital Communications Infrastructure (RDCI).

(D11) a duty to maintain accurate records of users (its customers) and maintain an efficient and effective channel of communication with each customer for the purpose of this conveyance.

If it is incumbent upon an OEM to know who its customers are, considerations of symmetry suggest that it is also incumbent upon the operator OP to know who its suppliers are. Maintenance of a suitable inventory is explicitly included in current international standards for IACS cybersecurity, the IEC 62443 series (Kobes 2018); however the first author considers that it is not always sufficiently clear what the properties of a sufficient inventory are, and has addressed this issue in (Ladkin 2017, Chapter 15: Devising and Maintaining System Inventories). The operator thus holds:

(D12) a duty to maintain an accurate and adequate inventory of software-based devices, sufficient for cybersecurity purposes.

We consider that deriving duties of a CSIRT, under current implementations a government authority, proceeds similarly from mandates, but is a somewhat broader matter than our limited scope here, which addresses IACS. An organisation such as a CSIRT can be constructed by government to fulfil certain specified duties which become its main functions, and it may not vacate these functions; whereas we are considering the case of organisations engaging in business round about an IACS, and to which duties accrue or shall accrue as a result of the business they choose to be in, which they can also leave if the duty structure is not amenable to them.

Duties incurred 'down the line' from vulnerability discovery and countermeasure development

Let us suppose now a legal environment in which the duties enumerated above are realised. Knowing about a vulnerability and its countermeasures, as above, is one thing; doing something about it is another. It seems to us reasonable that there should be duties concerning the implementation of such countermeasures, when countermeasures have been devised, post system deployment.

When an industrial plant is constructed by a system integrator SI, SI is in the first instance the customer of the OEM. When the plant P is operated by operator OP, the respondent to any issues concerning the equipment therein is OP. When a cybersecurity vulnerability V and countermeasures CM become known for a piece of equipment K, then:

(i) OP must come to know definitively whether P incorporates K or not.

(ii) If P incorporates K, OP must come to knowledge of V and CM.

(iii) OP must arrive in a situation somehow in which CM are implemented in P.

The first matter is covered by (D11): the OEM of K has a duty to inform OP; and OP has a duty to know it uses K. Similarly, the second point is covered by (D10), also carried by the OEM. The third matter is mandated by (M3) but a corresponding duty and holder have not yet been formulated.

We identify such duties and holders. Concerning knowledge of V and CM, in the first instance only the discoverer knows V, and then OEM and CSIRT know V and CM, as per (D5) and (D6). Under suitable confidentiality requirements, these would be the only parties with knowledge; that is covered by (D7). For OP to come to such knowledge, there must be a duty on one of those parties privy to that knowledge to convey it to OP. That is covered by (D11). So far, so covered.

We can also envisage the situation in which OP neither has the knowledge nor technical expertise to assess (or maybe even to understand) CM and whether they are implemented (correctly). Mandate (M3) requires the correct implementation, so here a party which does have that assessment expertise needs to be identified, and a corresponding duty formulated. Let us call this party the competent assessor (CAssessor). CAssessor might be OP; it might be OEM; it might be an organisation independent of either.

One solution would proceed via the following collection of duties. There is a defined user manual for K, which we may call UM(K), written by the OEM and available to OP. If K is modified by CM, call the resulting kit K+CM. Its corresponding user manual is UM(K+CM). We have already formulated in (D11) and (D12) duties for OEM and OP to know that P incorporates K, if it does. Here follow:

(D13) A duty, carried by OEM, to modify UM(K) to incorporate CM; let us say UM(K+CM).

(D14) A duty, carried by OP, to replace K with (K+CM) *if possible.*

(D15) A duty, held by OP, if it is possible to replace UM(K) with UM(K+CM), to do so.

(D16) A duty, held by OP, if UM(K) is replaced with UM(K+CM), to modify its procedures conformant with UM(K+CM).

So far so good. But it might not be possible per (D14) for OP to replace K with (K+CM). Even though OEM has developed K+CM to counter V, it may be inappropriate for OP to replace the kit K with K+CM; there may be dependencies on other equipment which rule it out, or concerns about such dependencies which may as yet be unknown. The situation becomes more difficult. It suggests that there should be two sets of countermeasures to be considered:

(i) K+CM, the upgrade to K;

(ii) CM-alt, countermeasures involving the environment of K, developed for the situation in which K cannot be seamlessly upgraded to K+CM.

There may be many such CM-alt developments, depending on the reasons an upgrade to K+CM might be hindered. For simplicity, we consider just one CM-alt. From this observation follows:

(D17) a duty to develop CM-alt in cases in which replacing K with (K+CM) is inappropriate. This duty is most plausibly carried by OEM, CAssessor and OP.

Restricting this duty to a specific subset of these three would lose some of the generality we are attempting to maintain. For example:

(i) if the vulnerability occurs in a proprietary part of K, then the OEM would be the only competent party to develop CM-alt, so it would be difficult here to see how OP and CAssessor could fulfil any duty to co-develop, except in a willingness to supply information as required by OEM;

(ii) the OEM may or may not agree with a determination by OP and/or CAssessor that it is inappropriate to install (K+CM) in P. If OP and/or CAssessor demand CM-alt, and OEM does not agree that CM-alt is necessary, then there must be a way of resolving that difference in judgement:

(a) either OEM will build CM-alt against its judgement, in which case OEM can reasonably expect to be suitably compensated for what it deems to be unnecessary work; or

(b) CM-alt will not be devised, OP is bound to incorporate (K+CM) and undergoes the knock-on effects of this in the rest of P, which OP and CAssessor deem deleterious and/or avoidable. Again, some form of compensation could be provided.

We imagine that such issues following difference in judgement can be resolved in the usual ways through regulatory arbitration, individual arbitration, or even litigation. For example, a jurisdiction may consider it desirable to require OEM to devise CM-alt on demand of OP and CAssessor (and/or the regulator). If so, some sort of indemnity protocol for supplying the resources required to do so could accompany such a legal requirement.

Some general duties do follow if OEM, CAssessor, and OP all agree that CM-alt is needed. In this situation, there is:

(D18) (conditional) a duty to determine whether K+CM should be implemented, or CM-alt, in cases in which both are available. This duty is most plausibly carried by CAssessor *along with* the regulator;

(D19) (conditional) a duty, carried by OEM, to develop CM-alt, should such alternative countermeasures be deemed to be necessary by a competent authority.

Depending on the jurisdictional choices, such a competent authority for (D19) could be CAssessor, a regulator, an arbitrator or a court. However, whether an OEM can develop effective and plausible CM-alt is surely dependent upon the exact nature of K and V. If, as assessed by CAssessor, alternative countermeasures (which most plausibly contains use limitations) cannot be devised that are as effective as upgrading to K+CM, then there is further:

(D20) a duty to determine what portion of V remains if, instead of installing K+CM, CM-alt is used instead. This duty is most plausibly carried jointly by OEM and OP. Part of this duty may comprise tasking CAssessor to determine this.

Given that part of V may remain after using CM-alt, it must be clear to all parties that this remaining part of V may be exploited by a malign third party. In case of such an exploitation of V, the question will arise: who is liable for damage caused in the case in which CM-alt has been implemented? Determining duties lies within the scope of this paper, but not determining liabilities. Given that determining liabilities for damage is a matter for arbitrators and courts of law, it would be inappropriate, as well as out of context, for us here to attempt to prejudge individual cases.

Possible duties concerning unidentified defects

It is possible that a defect remains unidentified until it is exploited and damage is incurred. In the software industry in general, there has developed a situation with 'shrink-wrapped' software, whereby the software developer warrants the software for absolutely no use or purpose whatever. It is intended that users shall have no recourse to the developer for any disadvantages or losses incurred through activation of defects in the software. We consider such a situation for safety-related critical infrastructure unacceptable, and software delivered with safety-critical equipment K from an OEM does not generally follow this paradigm (the SI purchasing K will insist on an OEM warranty of some sort, or look elsewhere. Indeed, the Siemens perspective (noted above) implicitly acknowledges such warranties).

However, as occurred in the Bookout/Toyota litigation,[6] it can be that extensive measures are taken to assess certain software for even a very specific behaviour, without it thereby being determined that that specific behaviour of the software is possible, even though it is. Such an assessment would be even harder if the search were for general dangerous behaviour of unspecified type. Such behaviour could well remain possible even under the limitations expressed in the user manual.

We believe that such a situation is avoidable, and as mentioned we have variously worked with regulators and standards organisations in order to propagate methods which enable objective properties of software, including some crucial to safety performance, to be established definitively during development. It is possible to develop software in such a way that, for example, 'run-time errors'[7] are completely eliminated (the authors know of two products/development methods on the market for which this claim is plausibly made). Since run-time errors are almost always associated with the halt of the software-based system in

[6] Bookout was the driver of a Toyota car in the US, who claimed it was subject to an unintended acceleration, which caused the car to crash, resulting in death and injury. The engine controller was digital, based on software. As a result of an extended investigation, including both NASA and court-appointed engineering experts, a civil court in Oklahoma found for the plaintiff Bookout, on the balance of probabilities. The engineering analysis involved was extensive, expensive and unprecedented in its detail. See (Ladkin and Thomas 2020) for some further details.

[7] A run-time error is a specific kind of program execution failure in computer science.

which the error occurred, such errors can be dangerous in an IACS controlling an industrial process which may subsequently behave in an unsafe way if the IACS halts. It is a matter of some wonder, and of some concern, to us that the requirement in English law to reduce risks as low as reasonably practicable (Health and Safety Executive 2001) has not (yet) resulted in pervasive use of these run-time-error-eliminating technologies in general safety-critical software development.

How might this situation be captured in mandates and duties? For elimination of run-time errors, one may formulate a duty on the developer so to eliminate them, for it is technically possible to use development methods which do that. For more general defects which express themselves in a behaviourally complex way, the task is similar, but technically more complex, to develop software demonstrably free of dangerous failure.

The organisation most obviously responsible for the quality of the software is the OEM – it is their product. None of the other parties that have an interest in a system explicitly mentioned in this paper can be considered reasonably to have any liability for the quality of the OEM's product (although they well may have responsibility for its assessment). In a context such as IACS in which the 'shrink-wrapped' agreement is evidently infeasible, and damage caused by a defect is incurred, the OEM shoulders the liability. Could there be cases in which damage is caused through a defect in a software-based system, deployed and implemented according to the user manual, in which the defect was unknown beforehand, and the OEM does not incur liability? We would argue not. It follows that the appropriate regime is one of strict liability (in the common meaning of the term in legal contexts).[8]

Consider, as an example, the following line of argument. Software written in a high-level programming language[9] such as C, C++, Fortran is prone to defects (Ladkin and Thomas 2020). This phenomenon can be addressed in part through a proposed mandate for improved development processes. There is a whole engineering discipline of Quality Management devoted to this. It can demonstrably be done for software, not only as above with respect to certain kinds of software defects but in general. We can formulate a possible mandate more precisely for software-based systems as follows:

(PM1) that:

(i) the properties that the system *must* have must be formulated explicitly and unambiguously, and

(ii) the operating conditions under which the system must exhibit these properties must be formulated explicitly and unambiguously, and

(iii) the required level of confidence that it does have those properties in those operating conditions must be explicitly formulated, and

(iv) evidence shall be provided, to the required confidence level, that the system architecture and design have those properties, and

[8] One might well consider strict liability here to be reasonably modified by the ALARP criterion of English law (Asquith LJ in *Edwards v National Coal Board* [1949] 1 KB 704), which is roughly that there is an obligation on the duty holder to reduce risk 'as low as reasonably practicable'. ALARP as practised is quite strict, with the burden of proof on the duty holder to show they have adhered to it.

[9] A programming language such as C, C++, Fortran is called a 'high-level language' (HLL) to distinguish it from assembler programming language or machine code. Machine code or object code are names for specific arrangements of bits which are directly executed by the processor. Assembler code is a more humanly-readable formulation of machine code, used for programmers to figure out directly what the processor is commanded to be doing. HLLs abstract from such details, and is at time of writing the usual way of writing programs. Programs in HLL are translated into object code by an automatic translator called a compiler; there are some other steps also involved in rendering an HLL program into machine code ready for execution. See (Ladkin and Thomas 2020) for some more detail.

(v) evidence shall be provided (to the required confidence level) that the high-level programming language (HLL) implementation and any pre-existing components have those properties.

This issue is faced also by the construction industry, in which there are also multiple parties involved in realising a given building or construction. The approach taken in that industry is that an architect designs the proposed solution at a high level, and then works with specialist engineers to refine the architecture into an implementable specification and design, that has been analysed and shown to have the required safety properties.

All of the measures noted above are known weak points in current high-reliability-software development, and all are addressed by effective development processes. Notice that we have not formulated specific duties; for example, we have not indicated which organisations have the duty to formulate the required system properties explicitly and unambiguously. Some of our colleagues think that it is the responsibility of the system commissioners (the client; the system integrator) to formulate such properties and simply give them as *fiat* to the software developer/OEM. Other colleagues believe that such a task can only reasonably be shared between the commissioners of the system, the OEM, and those who are expert in the techniques required to formulate the required properties (it currently requires a degree of expertise which is not yet widespread, and there is controversy over who should be required to exhibit such expertise and how). It is for a separate work to formulate our views on these matters.

The measures enumerated above for improvement of software quality will partially work, but not completely, because of the phenomena particular to software discussed in (Ladkin and Thomas 2020), in particular, the fourth phenomenon (unanticipated operational environment) and also because of the second phenomenon (the loose binding between the semantics of some high-level languages and the behaviour of the machine-code), as well as the branching nature of SW execution (many decision points, making it computationally infeasible to exercise them all in the operational context, no matter where you think responsibility may lie).

One can address these issues with further specific mandates regarding software development. Let us consider some of the options here. We designate them '(PM)' for 'possible mandate'. For example, one could propose:

(PM2) a mandate for unambiguous binding between HLL meaning and machine code, with requisite evidence provided.

Measures to implement most of (PM2) are available, even straightforward, but overall this is nowadays still a difficult task. The operational context which is necessarily fed in during translation (compiling and linking) is unsurveyably complex.

We might thus consider further:

(PM3) a mandate for explicit restriction of the operational context ('environment of the program') to fall within the domain of validity of the measures expressed in (PM1) and (PM2).

The one phenomenon for which we (and many experts) see no evident solution is the fourth: to ensure the pertinent operational context has been described completely. These three mandate suggestions correspond to the following potential duties:

(PM1) has as duty holder the programming house.

(PM2) has potentially multiple duty holders:

(i) the designers of the HLL;

(ii) those providing translation tools (compilers and libraries) for it;

(iii) the hardware designers and their machine code.

(PM3) has also multiple potential duty holders:

(i) the providers of the machine code (which must be those listed under (PM2));

(ii) someone, maybe amongst those listed for (PM2) but maybe someone else, responsible for explicitly specifying the operational context in which that machine code is used (undetermined as yet);

(iii) SI, who must determine whether the actual operational context of the machine code fulfils the operational-context-specification.

It is beyond the scope of the present paper to propose from these alternatives a concrete solution to the software-quality issue for IACS. However, we do see improving the quality of software, if necessary through legislation, as essential to the improvement of safety and cybersecurity in all installations, not just in IACS.

Summary

We considered software-based components of IACS. We referred to the quality of software, which is less dependable than many physical engineering artifacts. We considered mandates, namely requirements for a specific sort of action which follow from the organisational structure of process plants deemed to be part of critical infrastructure, namely SI, OEM and OP, as well as the organisation mandated by the NIS regulations, namely CA and CSIRT. We considered one industry view on IACS cybersecurity responsibilities. Considering the nature of software, along with the traditional roles commissioning, integrating and operating industrial plant, we then formulated some high-level mandates (general obligations on unspecified organisations), and duties which follow from those mandates, namely obligations on specified organisations (the duty holder), in the context of a CSIRT infrastructure. We showed the efficacy of this approach by deriving duties further 'down the line' in integration, deployment and operation of IACS.

Finally, we considered the situation with unidentified defects. We acknowledged that the situation concerning improved development methods for software and mandates for their use is (still, and in our view unfortunately) industrially controversial. As a basis for continuing discussion of the software quality issue, we formulated some 'proposed mandates' which in our view would facilitate the pervasive introduction of such improved methods.

Prof. i.R. Dr. Peter Bernard Ladkin is a systems-safety specialist with a background in software dependability and logic. His causal accident analysis method Why-Because Analysis (WBA) is used by some 11,000 engineers worldwide. He taught at Bielefeld University and heads tech-transfer companies Causalis Limited and Causalis Ingenieurgesellschaft mbH.

Martyn Thomas CBE FREng is Emeritus Professor of IT at Gresham College. He has worked in the software industry for 50 years, closely involved with Government departments and university research, as a non-executive director of the Health & Safety Executive and was an expert technology advisor to the National Air Traffic Services (NATS). He has been an expert witness in major litigation and held visiting professorships at Oxford, Aberystwyth, Bristol and Manchester.

References

Böllmann, Sören 2018. Personal communication with the first author, February 2018.

CISA (no date) https://www.cisa.gov/uscert/ics

de Haan, Johannes, Peter Bernard Ladkin and Bernd Sieker, 2016. Coping with Wide-Scope ATC/ATM Communications Failure. White paper, RVS Group, University of Bielefeld, March 2016.

EC 765/2008. Regulation (EC) No 765/2008 of the European Parliament and of the Council of 9 July 2008 setting out the requirements for accreditation and market surveillance relating to the marketing of products and repealing Regulation (EEC) No 339/93 (Text with EEA relevance), *OJ L 218, 13.8.2008, p. 30–47*

EU 2016/1148. 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, *OJ L 194, 19.7.2016, p. 1–30*

Health and Safety Executive, 2001. Reducing Risks – Protecting People. HSEBooks. Available from http://www.hse.gov.uk/risk/theory/r2p2.pdf

International Electrotechnical Commission (IEC), 2010. IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements, Edition 2.

Kobes, Pierre, 2018. Personal communication with the first author, 24 April, 2018.

Ladkin, Peter Bernard, 2017. A Critical-Systems Assurance Manifesto, eBook, RVS-BI Group, Bielefeld 2017. Available from https://rvs-bi.de/publications/RVS-Bk-17-01.html

Ladkin, Peter Bernard and Thomas, Martyn, 2020. Software Quality, Its Nature, and the Cultures of Building It, preprint, ResearchGate, DOI: 10.13140/RG.2.2.27806.89925, March 2020 (original version 2018). Available from https://www.researchgate.net/publication/339697246_Software_Quality_Its_Nature_and_the_Cultures_of_Building_It

Ladkin, Peter Bernard, Littlewood, Bev, Thimbleby, Harold and Thomas, Martyn, The Law Commission presumption concerning the dependability of computer evidence, Digital Evidence and Electronic Signature Law Review 17, 2020. Available from https://journals.sas.ac.uk/deeslr/issue/view/578

Siemens 2017. Siemens AG, Regulation and Conformity Assessment of Security Relevant Products and Solutions – a Siemens Perspective. August 2017.