The Emergence of Accidental Autonomy

Alastair Faulkner† and Mark Nicholson‡

†Abbeymeade Limited, ‡University of York

Abstract – The Boeing 737 MAX - Manoeuvring Characteristics Augmentation System (MCAS) accidents have demonstrated how cumulative factors may lead to accidental autonomy. Accidental autonomy emerges when differences in models compete over resources and control. In the operational domain, one manifestation is failure at the human-machine interface. Subtle, incremental changes in technology allied with downward economic pressures encourage reuse to create the system safety property of additionality. Cumulative incremental changes occur that when taken together, are safety significant. Reuse of process, product or both gives rise to inappropriate design trade-offs. Assumptions about the completeness of process, design, implementation or context may lead, in extreme circumstances, to the creation of accidental autonomy - systems without human oversight that implement safety-related functionality or services.

Oversight, assessment and approval of systems dependent on reuse are reliant on the familiarity of the assessor with the reused elements within their operational and use context. Incomplete, inadequate understanding and failures of comprehension, along with the allure of fast software development, create the potential for accidental autonomy.

1 Introduction

Systems engineering is subject to several capability and economic pressures. This has driven Systems Engineers to create systems from generic components. To become generic components are designed to depend on data to be configured, characterised and parameterised to the required behaviour. This data dependency is also evident in the broader system, its subsystems, interfaces and shared overlapping datasets.

Our collective understanding of autonomy [1] is bound to context [2] and era. It is context that gives legitimacy to a person or systems actions, the facts or processes of doing something, typically to achieve an aim. [3] Automation often first appears as an aid to support existing practice. These aids are developed and evolve to support and reinforce changes – driving uniformity and consistency, but not necessarily improving reliability, performance or resilience. [4] The overall impact on the working environment is a dramatic increase in the volume of digital

data and flow of that digital data around system elements and a concomitant decrease in understanding of the context and limitations of this autonomy. [5]

The potential for accidental autonomy arises when changes are implemented as 'islands' of functionality to support identified activities. We use the word accidental [28] as happening by chance, unintentionally, or unexpectedly. Existing system interfaces characterise the boundaries of these new functionalities. Operators become reliant on the autonomous actions of systems (and its assumed functional model). Even if they can intervene, they often do not, as they tend to trust the technology.

At the same time, a reliance on automated decision-making increases. At the lowest level of autonomy, [6] computers offer no assistance; they facilitate information acquisition. Later, they offer a set of decision alternatives, facilitating analysis. They provide increasing amounts of support for the decision-making process itself. The operator has a restricted time to overrule the autonomous decision before an action. Finally, the highest levels of autonomy provide implementation with no capability for the human to overrule and little, if any, information provided on what actions the autonomy has undertaken.

Safety risk arises where the operators understanding of the system and the role of autonomy is incomplete. A classic human-machine interaction failure may result. Clean design and efficiency are frequently used to justify autonomy. Autonomy may correct an underlying instability in the system model or design. It becomes accidental autonomy when its actions arise from incomplete design, implementation or its use is outside an acceptable design envelope with respect to safety. Further, it may induce additional human failures due to mismatches between the human mental model, the goals of autonomy, and what is detected, by both parties, of the real-world context.

Systems Engineering has progressed to the point where machines have the capability to undertake high-level decisions and enact consequent actions without recourse to direct human input. As a result, we should not assume the presence of a user; instead, we employ the term actor as 'an individual, entity, or combination of product (including autonomy), people, and process'. This raises the question of supervision [11], and to what extent humans remain involved in operational decisions.

2 System as the Fundamental Concept

A system is a (purposeful [7]) set of things working together as part of a mechanism or an interconnecting network; a complex whole. [8] Systems operate in increasingly open environments. These environments and the data exchanged within them are homogeneous or heterogeneous or a mixture of both. The nature of the environment influences the formation of the system boundary (porous or secure (as defined by an appropriate security model)). For modern complex systems, it is common to present several views of the system either across many

sheets (possibly in a hierarchy) or to separate physical realisations from abstract (logical) models. Therefore, reviewing any classification system requires an appreciation of the context, including the role of the actors within each viewpoint.

Systems of Systems (SoS) bring together a set of systems for a task that none of the systems can accomplish on its own. Each constituent system keeps its management, goals, and resources while coordinating within the SoS and adapting to meet SoS goals. [9] Interconnected SoS give rise to accidental tasks associated with the mapping and representation of abstract entities and mapping of those entities onto the constraints of the solutions. [10] One implementation uses Internet of Things (IoT) devices as generalised platforms, enabling them to be adapted and configured to a range of solution areas. Often these include a highly capable Operating System (OS) that can provide a full range of communication and computational services. They are configured (and characterised) to a particular task (or range of tasks) through data. An additional dimension to SoS would be to add (joiners) or remove (leavers) to the system. A joiner introduces additional capabilities, capacities and tasks. A leaver removes them. The system is modified (adapted) to reflect these changes in the service catalogues.

There are several interrelated models at play here for a complex system.

- Security – one starting point requires all system entities to be assigned an identity so that an actor's access privileges can be assigned (and revoked). This leads to consideration of the relative authorities between the system and the actors that use and are served by the system.
- Maintenance – it is common to use Permit to Work (tickets) as part of a formal maintenance procedure to isolate physical plant and equipment. How should autonomy and their respective services be controlled while maintenance is being enacted? How will the system be reconfigured to manage reduced capability whilst one or more systems (and their associated (and (mutually) dependent services)) are maintained?
- Operational – how big, complex or complicated should a system be before the risks associated with its failure demand the creation of an operational model? It is common to consider operational modes during system safety management activities. An operational strategy should be used to direct and inform the creation and maintenance of the operational model. Issues associated with size and complexity require that a decision model is constructed and maintained over the operational model.
- Safety - in an ideal world, control and protection function differentiation would be applied universally to function, flow and service. The desirable safety features of hardware and software are well established. It is not clear that such requirements are applied to services. Issues associated with size and complexities require that a services model is constructed and maintained.
- Risk - in systems which are dependent on autonomy, the form and nature of risks are multi-dimensional, crossing many discipline and system boundaries. Many of these boundaries are indistinct. The risks associated with reliance on autonomy require the reappraisal of existing risk models.

- Supervision – the action of supervising someone or something. [11] A supervisory model is a scheme for specifying and enforcing supervisory policies.

The use of SoS and IoT technologies means that an integrated risk model across these models is required to address residual and unsecured functionality.

3 Autonomy

Autonomy [1] is not new. It is used to describe human political activity, for a region 'having its own laws'. In the modern sense, autonomy is readily adapted to address systems capable of operating without direct human control [12] but varying degrees of human supervision or oversight. At one extreme automaton [13] is confined to actions described by a predetermined set of coded instructions. At the other are learning systems that adapt their behaviour in response to changes in the operating environment – its context. [2]

It is context – 'the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood' [2], that provide the basis of this paper. Context is not limited to the operational environment. For the engineered system, it reflects the designer's expectation of the operational environment. This position is further complicated by what learning systems 'understand' as the basis for the formulation of its response.

Consider an automatic [14] washing machine; it works by itself with little or no direct interaction. The use or introduction of multiple automatic devices into an existing context creates automation. [15] These definitions contain an implicit expectation of a static context. That, the context is known, and if it changes at all, it changes slowly under controlled conditions. What if, a new generation of automatic devices, providing a form-fit-function [16] replacement, are introduced based on IoT. Suppose the system has unused capability and capacity. An unsecured 'discovery' function recognises other IoTs and connects to this residual and unsecured functionality. Taking a SoS perspective, this represents one or more emergent properties [17], possibly with unintended consequences.

Autonomy becomes multi-dimensional under Industry 4.0: [18, 19]

- The vertical integration of flexible and reconfigurable systems within businesses;
- The horizontal integration of inter-company value chains and networks;
- The product life-cycle integration of digital end-to-end engineering activities across the entire value chain of both the product and the associated systems.

3.1 Accidental Autonomy

In a connected system, automation creates a dramatic increase in the volume of digital data and flow of that digital data around system elements and a concomitant decrease in understanding of the context and limitations of that data. Change is a crucial factor in creating the potential for accidental autonomy. At one extreme are revisions of an existing product or model, as with an aircraft. At the other are a series of incremental changes in the pre-existing operational context. Existing boundaries may not constrain these new functionalities. An actor may become reliant on data produced by another actor (passed across one or more boundaries), with little ability to influence the data stream they have become reliant on. At the same time, reliance on automated decision-making increases.

Accidental autonomy results when differences between models of use, and context of use, are not sufficiently well understood in all operational modes. This includes misuse. The true nature of its inclusion in the system is omitted, or downplayed, in the safety ensurance and assurance process. As a result, insufficient safety mitigations are provided, and poor human-machine interactions may occur. We can conceive of accidental autonomy arising between two or more elements of a system, perhaps as a complete system, or SoS. Within any given context, these elements, systems or SoS have different responsibilities, of which some will be safety-related. The accidental autonomous system could co-exist with people-centered activities reliant on a predefined set of processes. A fundamental assumption is that the people within the system are trained, competent and experienced enough to deliver the required operation (including its safety management); this implies a maturity of definition and application. Therefore, the provision of the product or process is dependant on context.

Extending the concept of emergent properties [17] gives rise to the concept of emergent autonomy. Emergent autonomy is a consequence of the interactions and relationships between system elements rather than the behaviour of individual elements. [7] The nature of emergent autonomy is linked to robustness and resilience and is a critical contribution to safety. Accidental autonomy is a subset of emergent autonomy and may be a result of incomplete development activities. Consider an existing design. This design has been in production and operation for many years undergoing successive revisions. Each revision ‘refreshes’ the technology, typically the control systems. The effects of seemingly minor changes become cumulative, giving rise to the safety property of ‘additionality’. Budgetary, time and project management constraints limit the safety analysis to a subset of changes. Given these conditions, it is easy to see how the effects of automation and increasing levels of autonomy are overlooked. For example, an aircraft design will be revised over many generations of the airframe. It is not unusual for the aircraft model to evolve over 40 years as with Nimrod (as an extensive modification of the de Havilland Comet). We await, with interest, the two accident reports for the Boeing 737 MAX.

For visualisation, we reuse elements of [4]. Fig. 1 illustrates the footprint of accidental autonomy.

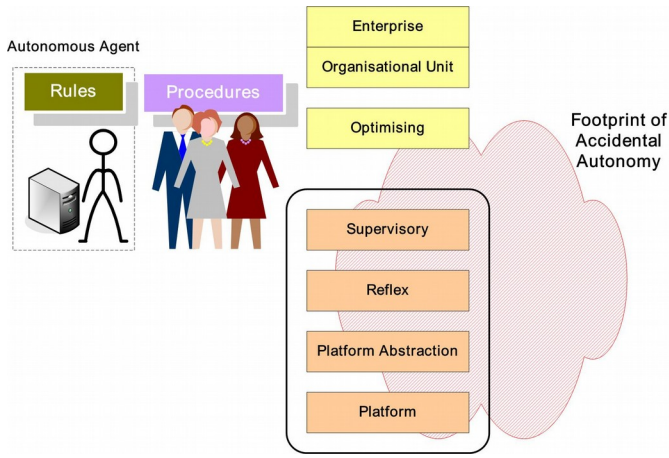

Fig. 1. A layered model for a hierarchy of systems and the footprint of accidental autonomy

How far up the hierarchy could accidental autonomy reach? Potentially, all the way to the top. Decision support systems are ever more reliant on data analytics, data science, data engineering and autonomy. Fig. 1 also illustrates the supervision within the hierarchy.

As autonomy moves through the hierarchy of abstraction, its responsibilities and authorities change. Similarly, the ability of humans to provide Safety-I [20] mitigation procedures needs to be addressed. Improved diagnostics, monitoring and higher-level response, are required. This challenges the ability to design efficient procedures. Furthermore, it impacts on the ability of humans to execute Safety-II [20] dynamic mitigations, as their mental model of the system and how they interact with it is flawed.

The implementation strategy must include a means to impose a boundary to the propagation of the actions of the system and the impact of failures on the availability and safety of the system. Fig. 2 illustrates how Interface Agreements (IA) [5] provides that functionality for new and legacy systems.

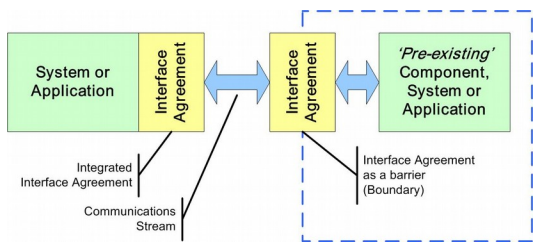

Fig. 2. Implementation Model for Interface Agreements

A series of interfacing elements can be envisaged. Transformations occur as the sequence is progressed. One or more actors supervise this series of transformations. These chains, and associated transformations can occur at multiple levels of abstraction. As a result, a number of different aspects to Autonomy can be identified. Any one of these organisational aspects may lead to developers creating accidental autonomy.

3.2 Vertical Autonomy

Vertical expansion is used to describe an organisation that grows through the acquisition of companies that produce the intermediate goods needed by the business. Economic pressures create management and organisational structures that become more rigid and inflexible. These companies become monolithic, often creating closed implementations in silos. Over time this inflexibility makes it difficult for a company to respond to changes in the marketplace. Implementations based on SoS and IoT offer an opportunity to break free from vertical silos. This requires vertical integration of flexible and reconfigurable systems within businesses.

These structures also apply to systems. Consider a basic control system that consists of input-controller-output. In past implementations, the input would be wholly dependent on the physical properties of the sensor. For example, a bimetallic strip is used to implement the function of a thermostat where specific temperature causes the differential expansion to deflect enough to close (or open) the contacts. These devices, once physically co-located, are now implemented using IoT on remote networks. They may even be replaced by more generic devices that sense a number of properties. The required data is then mined from the output of these sensors.

Here we use the following classifications of vertical autonomy:

- backward (upstream)
- forward (downstream)
- balanced (both upstream and downstream)

Fig. 3 illustrates vertical integration; supplier, manufacturer and distributor.

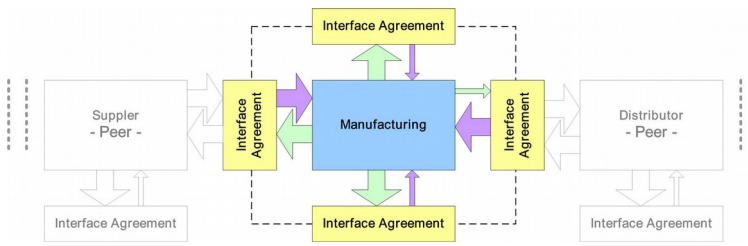

Fig. 3. Supplier, Manufacturer and Distributor

3.2.1 Backward (Upstream)

A manufacturer implements upstream expansion by purchasing a parts supplier. Upstream autonomy provides more data, command and control of the upstream systems. For example, accidental upstream autonomy contains an operational model, that when a candidate production schedule is interpreted, causes it to re-order stock for a future production run without seeking authority (or confirmation) that the production run would take place. Therefore, accidental upstream autonomy is consumption-led and may lag demand. In our control system example, upstream autonomy is where the controller reduces the frequency of updates from the sensor becoming less responsive.

3.2.2 Forward (Downstream)

A manufacturer implements downstream expansion by purchasing the distribution and sales network. Downstream autonomy allows the manufacturer to only produce what can be sold. Accidental downstream autonomy might misinterpret demand to produce too much or too little. Therefore, accidental downstream autonomy is demand-led. Sudden (step) changes in demand may create instability and lead to over and under production. In our control system example, downstream autonomy is where the controller checks the actuator (output), continually monitoring the energy required to assert the output has occurred. The demand is the energy (effort) required to assert the output.

3.2.3 Balanced (both Upstream and Downstream)

A manufacturer implements balanced expansion by purchasing a suppliers, distribution and sales network. Balanced autonomy contains a balance of consumption, demand and ‘damping’ (stabilising) elements. Accidental balanced autonomy may implement complex functions analogous to the Proportional-Integral-Derivative (PID) pattern in control theory. [21] Other patterns also apply. In our control system example, balanced autonomy is where the controller adapts to the operational requirement adjusting the input sensor rate to the best fit to the PID setpoint and deviations from it. At the same time, the controller calculates a predictive output anticipating the effort required to apply the output.

3.3 Horizontal Autonomy

A manufacturer implements horizontal integration through changes in capacity and capability. For example, the introduction of new technology replaces fixed function machines (manufacturing cells) using reconfigurable workstations clus-

tered in super-cells into a production line. This creates higher capacity and requires integration of the value chains and networks. In our control system example, horizontal autonomy combines an array of identical *balanced autonomy controllers*. Horizontal autonomy uses sensor fusion over the input devices and load-balancing across the outputs. This implies the use of supervision [11] over the array of controllers.

The above characterisation implies that horizontal data transformation and vertical abstraction transformations (along the lines of fig. 1) need to be considered when identifying Critical Control Points (CCP) to ensure that development processes do not introduce accidental autonomy. [22]

Paths (physical product or data) will incorporate CCP and may involve multiple sources and multiple sinks. These issues are compounded when these paths are dynamic. This dynamism is not limited to changing numbers of sources or sinks or processes but also changes in demand (capacity) and availability. Paths may be transient synthesised on demand for single-use and then discarded. Incident investigation is eased where these CCPs provide controls and logged data. Therefore, CCPs provide evidence about the actions of the system, including accidental autonomy.

To implement dynamic paths, one analogy would be to use a standardised library of elements across a node and link network. This provides one means of implementing redundancy where nodes are unavailable. In network theory, links can be assigned a weight; path management is used to identify a route with the least weight. Accidental autonomy need not be persistent; it may arise from transient elements of the dynamic formation of the system. Therefore, implementation requires the definition of an Identity Model and Security Model.

3.4 Product-line Autonomy

A product line can be defined as a set of systems sharing a common, managed set of features that satisfy specific needs and are developed from a common set of core assets in a prescribed way. [23, 24] A product-line [25] development employs a life-cycle model and the process it contains to develop the system definition, into the system. The defined set of features can be reused within defined fit-form-function [16] contexts.

Accidental introduction of Autonomy via product lines presents two potential threats; that the fit-form-function replacement introduces residual and unsecured functionality, and secondly that the system has grown organically and cannot support digital end-to-end engineering activities with a reasonable certainty of outcome. This may lead to the accidental autonomy being presented with a range, sequence and timings in an ‘unfamiliar’ environment that it cannot manage safely. The autonomy cannot rely on the human to retrieve the situation.

4 Cyber Physical System Threats

When deciding what steps to take to prevent and respond to threats, we might immediately focus on the threat of hacking. However, the range of threats systems face is much broader than this, encompassing anything that can adversely affect their operation, including theft, destruction, disclosure, modification or unauthorised access.

We modify the definition of threat [26] to be 'an actor likely to cause one or more hazards.' This definition includes autonomy within the actor. Changes in system context require resilience and robustness from the autonomy to withstand threats arising from identity, security and sneak attributes.

4.1 Identity

'Identity' should be a unique labelling of attributes of the object (system resource) being accessed and of the actor requesting access in a given context. Threats arise from identity error, duplicate and missing identities. A malicious, deliberate identity-based (spoofing) act could be used as a means to gain control of the system. One means to counter identity-based threats is the use of a formalised and managed 'identity model'. An identity model is a scheme for specifying and enforcing identity policies. An identity model is a key aspect of the security model.

Both emergent and accidental autonomy are sensitive to error, omission or duplicate identities, especially in dynamically reconfigurable systems as changes in system behaviour presents significant system safety management challenges. These issues are compounded where a system uses joiners and leavers as one means to satisfy operational demand, including capability and capacity. It cannot be assumed that identity theft applies only to users as it applies equally to actors, systems, assets, data and data paths.

4.2 Security

A security model is a scheme for specifying and enforcing security policies. A security model uses a formal model of access rights. Authorisation is implemented using identity and enforced through authentication. Potentially, failures of cybersecurity provide the intruder with unbridled access to a system. Identity and its management is a critical feature of both Data Safety and information security.

Should each instance of autonomy be required to be assigned its own unique identity? Low-level systems often do not implement a security or identity model. Autonomy in such systems can act without authorisation, often acting with 'super-

user' rights. More extensive systems required security models, and therefore, each actor or autonomy requires one or more identities.

4.3 Sneak Attributes

Systems may contain sneak (or hidden) attributes that may cause unwanted action or inhibit desired functions. [27] Sneak attributes arise where the physical realisation contains many more characteristics than the logical representation. Examination of simple network switches reveals capabilities to separate network traffic using configuration data. Errors in the configuration may permit 'mixed network' traffic, compromising the intended separation, and creating additional paths between entities.

These may include:

- Sneak paths: unintended paths within a system and its external interfaces.
- Sneak timing: unexpected interruption or enabling of a [function or service] due to timing problems which may cause or prevent the activation or inhibition of a function [or service] at an unexpected time.
- Sneak indications: undesired activation or deactivation of a [status] indication which may cause an ambiguous or false display of system operating conditions.
- Sneak identity: incorrect or ambiguous identity of a [function or service] which may cause actor error through inappropriate control activation.

As complex networks of autonomous actors embedded within systems emerge the ability to create accidental autonomy via a sneak, increases.

5 Discussion

There are many examples of automatic systems, from the washing machine to automobile automatic transmissions. The degree of possible automation increases by using SoS and IoT technologies. Economic pressures to increase efficiencies, such as fuel economy, drive change to the foundations of existing designs and the organisations that develop and operate them. The increased reliability, availability, capability and real-time response of control systems allow the designer to explore inherently unstable designs. These unstable designs offer potential operational efficiencies. This involves a design change from stable towards the edge of instability, where additional means are required to stay within the stability envelope. One means to achieve this is autonomy. Accidental autonomy contributes to the emergent property of incremental additionality. It may be unintentional, or unexpected, but it does not happen by chance. These emergent properties and au-

tonomies are systematic. Their behaviours are repeatable and may be inclusions (impurities) from an incomplete development.

This paper has outlined how autonomy is multi-dimensional. It follows that accidental autonomy is also multi-dimensional. The system safety question, 'how could this possibly go wrong?' is more relevant than ever. Early indicators of the Boeing 737 MAX accidents shows how organisational structures can interact with economic and engineering factors to create the potential for accidental autonomy. They illustrate how changes to a pre-existing model create a change in context where the users (pilots) are unaware of the underlying nature of change. One approach to addressing accidental autonomy is to increase the ability of pilots to address the unexpected. This will require them to become experts in diagnosing and addressing such issues. This higher competency is required to detect, diagnose and formulate a course of action during the operational event. This assumes that the actions of the user can result in a positive outcome. The more dimensions autonomy occupies, the more extensive, the more difficult - real-time - diagnosis becomes. We can no longer rely on the steady-state being a safe condition. Economics and human factors knowledge imply that this is not a credible approach. As a result, organisational and technical means must be found to identify and address potential accidental autonomy issues.

For all forms of autonomy, the permutations of threats, failures and latent hazards may be extensive but are foreseeable. Accidental Autonomy may result in unintended consequences. Merton [29] asserts that these are outcomes that are not the ones foreseen and intended by a purposeful action. The operational domain includes maintenance. What provision should autonomy make to include the statutory requirements for 'Permit to Work' (PtW) and its required 'Safe System of Work' (SSoW) based on one or more 'Safe Method of Work' (SMoW)?

Its hidden nature characterises accidental autonomy. Its use to manage the properties of an underlying design without adequate annunciation and user involvement contributes to the confusion of an ongoing incident. Its actions cannot be assumed to be benign or malevolent; they will be incomplete, in accidental autonomies pursuit of ill-defined, unknown goals.

6 Conclusion

No single approach resolves the difficulties associated with the essence [9] of engineered and accidental autonomy - those parts concerned with the fashioning of abstract conceptual structures of high complexity. Greater scale, scope and complexity give rise to an urgency to create strategies to manage the large-scale application of techniques and measures. In part, this urgency arises from the reliance placed on these systems and safety risks associated with their failure. Many systems are reliant on this connectivity and provide substantially reduced functionality when the interconnectivity fails. In contrast, previous generations of system

implementations operated as islands, separated and protected from external influences – and in that sense, self-reliant.

The first step in addressing accidental autonomy is recognition of its potential scale, scope and complexity. It will introduce new failure mechanisms due to differences in its required and the actual context. It is a multi-dimensional problem occupying vertical, horizontal and product-line axes. Its management will require many interrelated approaches and their associated techniques and measures. Its independence also provides new forms of latent failures. For example, two or more learning autonomous elements may adapt in different ways to changes in their operational environment. The new behaviours may introduce conflict and cause instability over many learning cycles. There is no guarantee that these differences will resolve into a stable state. Such variations will only be manifest in an incident.

7 References

1. Autonomy – "freedom from external control or influence; independence" www.lexico.com/en/definition/autonomy (visited 10 October 2019)
2. Context "The circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood." www.lexico.com/en/definition/context (visited 10 October 2019)
3. Action - "fact or process of doing something, typically to achieve an aim", www.lexico.com/en/definition/action (visited 10 October 2019)
4. Alastair Faulkner and Mark Nicholson, "An Assessment Framework for Data-Centric Systems", Proceedings of the Twenty-Second Safety-Critical Systems Symposium, Brighton, UK. Edited by Chris Dale and Tom Anderson. ISBN 978-1491263648. Safety Critical Systems Club, 2014
5. Alastair Faulkner and Mark Nicholson, "Data-Centric Safety: Challenges, Approaches, and Incident Investigation", Elsevier, to be published March 2020, ISBN: 978-0-12-820790-1
6. R. Parasuraman, T. B. Sheridan, and C. D. Wickens. A Model for Types and Levels of Human Interaction with Automation. IEEE Transactions on Systems, Man, Cybernetics—Part A, Vol. 30, No. 3, 2000
7. P. Checkland. Systems Thinking, Systems Practice. 1981 John Wiley & Sons
8. System –"A set of things working together as parts of a mechanism or an interconnecting network; a complex whole." www.lexico.com/en/definition/system (visited 10 October 2019)
9. BKCASE Governance and Editorial Board. Guide to the Systems Engineering Body of Knowledge (SEBoK). 2017
10. Frederick P. Brooks. No Silver Bullet — Essence and Accident in Software Engineering. Proceedings of the IFIP Tenth World Computing Conference, 1986, pages 1069–1076

11. Supervision - “action of supervising someone or something” www.lexico.-com/en/definition/supervision (visited 10 October 2019)
12. Autonomous - “denoting or performed by a device capable of operating without direct human control.” www.lexico.com/en/definition/autonomous (visited 10 October 2019)
13. Automaton – “machine which performs a range of functions according to a predetermined set of coded instructions” www.lexico.com/en/definition/automaton (visited 10 October 2019)
14. Automatic – “(of a device or process) working by itself with little or no direct human control.” www.lexico.com/en/definition/automatic (visited 10 October 2019)
15. Automation – “use or introduction of automatic equipment in a manufacturing or other process or facility.” www.lexico.com/en/definition/automation (visited 10 October 2019)
16. Morris, R. The fundamentals of product design. AVA Publishing. 2009
17. SEBoK (Guide to the Systems Engineering Body of Knowledge): Emergence, www.sebokwiki.org/wiki/Emergence (visited 10 October 2019)
18. David Leal-Ayala, Jennifer Castañeda-Navarrete and Carlos López-Gómez, OK Computer? -The safety and security dimensions of Industry 4.0, Global Manufacturing and Industrialisation Summit (GMIS) and Lloyd’s Register Foundation (LRF). 2019
19. Mario Hermann, Tobias Pentek and Boris Otto, “Design Principles for Industrie 4.0 Scenarios” in 49th Hawaii International Conference on System Sciences (HICSS), pages 3928-3937, 2016
20. Erik Hollnagel. Safety-I and Safety-II. ISBN 978-1472423085. Routledge, 2014
21. Araki, M. "PID Control", Control Systems, Robotics and Automation, Vol II, 2011
22. Mark E. J. Newman. Networks: an Introduction. Oxford, 2010
23. Paul Clements and Linda Northrop. Software Product Lines: Practices and Patterns. Addison-Wesley Professional, 2001
24. Linda Northrop and Paul Clements. A Framework for Software Product Line Practice. Software Engineering Institute, 2012
25. André de Oliveira et al. Supporting the Automated Generation of Modular Product Line Safety Cases. Volume Theory and Engineering of Complex Systems and Dependability. ISBN 978-3-319-19215-4. Springer International Publishing, 2015, pages 319–330
26. Threat - “A person or thing likely to cause damage or danger.” www.lexico.-com/en/definition/threat (visited 10 October 2019)
27. MIL-STD-785B Reliability Program for Systems and Equipment Development and Production. US Department of Defence, 1980.
28. Accidental – “happening by chance, unintentionally, or unexpectedly” www.lexico.com/en/definition/accidental (visited 10 October 2019)
29. Robert K. Merton "The Unanticipated Consequences of Purposive Social Action" American Sociological Review. 1 (6): 895. 1936