Making Enterprise Information Systems Resilient Against Disruptive Events: A Conceptual View

Boris Shishkov[1,2,3(✉)] and Alexander Verbraeck[4]

[1] Faculty of Information Sciences, University of Library Studies and Information Technologies, Sofia, Bulgaria
[2] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria
[3] Institute IICREST, Sofia, Bulgaria
b.b.shishkov@iicrest.org
[4] Faculty of Technology, Policy, and Management, Delft University of Technology, Delft, The Netherlands
a.verbraeck@tudelft.nl

Abstract. *Enterprise Information Systems* (*EIS*) are designed to deal with *normal variability* in their inputs and data. Empowered by CONTEXT-AWARENESS, some *EIS* even count on *sensors* and/or *data analytics* for capturing changes outside of the system. Nevertheless, *context-awareness* would often fail when *EIS* are affected by (large-scale) *disruptive events*, such as disasters, virus outbreaks, or military conflicts. Hence, in the current paper, we take a step forward, by considering *context-awareness* for *disruptive events*. We combine *context-awareness* with *risk management* techniques, such as FMECA and FTA, that are useful for defining and mitigating risk events. To avoid having to define the likelihood for such very-low-probability disruptive risks, we use CONSEQUENCE-BASED RISK MANAGEMENT rather than traditional *risk management*. We augment this approach with the *context-awareness* paradigm, delivering a contribution that is two-fold: (i) We propose *context-awareness-related measures* and *consequence-based-risk-management-related measures*, to address *disruptive events*; (ii) We reflect this in a *method featuring the application of context-awareness and risk management for designing robust and resilient EIS*.

Keywords: Enterprise information system · Resilience · Context-awareness · Risk management

1 Introduction

Larger organizations are essentially supported by *Enterprise Information Systems – EIS* [1, 2], such as *Enterprise Resource Planning* systems (*ERP*), *Customer Relationship Management* systems (*CRM*), and *Supply Chain Management* systems (*SCM*). Such systems help organizations' *business processes* to run smoothly and to be of full value [3]. Their correct working assumes an adequate alignment between the *EIS* and the enterprise *environment* [4].

B. Shishkov (Ed.): BMSD 2020, LNBIP 391, pp. 38–54, 2020.
https://doi.org/10.1007/978-3-030-52306-0_3

As the environment of organizations is continuously changing, such an alignment can only be achieved if the *EIS* is *situation-aware* – this means sensitive to environmental changes [5]. *Sensors* [6] and *data analytics* [7] provide information about the state of the system and its environment, and *EIS* can adapt to perceived changes using *run-time behavior algorithms* [5].

Under regular business uncertainty, this is supposed to work. Nevertheless, when *large-scale disruptive events* occur, adaptive algorithms stop working. For example: (i) A virus outbreak or a large-scale strike in a country may effectively "shut down" businesses, public services, and logistics [8]; (ii) A disaster may physically destroy computer assets of partner organizations such as suppliers and customers [9, 10]; (iii) A cyber-attack may cause huge disruptions in the technology that supports business processes [15].

As a result, organizations that run an *EIS* and cannot adapt sufficiently, would essentially stop functioning. This is because of the dependence on their *EIS* that cannot deal with the exceptional changes in the current situation, providing sub-optimal support to corresponding business processes. It seems that businesses are not prepared to act in such situations [8] and have to fall back to manual interventions for which the IT-supported business processes are not designed.

We argue that what is needed during disruptive events is a "*resilient mode*" for *EIS*, building on the following four characteristics:

- The *EIS* determines when the data is out of bounds by setting boundaries for parameters in the environment and scanning the environment for parameters that don't fall into the boundaries (context awareness);
- The *EIS* has ways to fall back to atomic processes that are less integrated than the processes that are normally carried out, and that can temporarily deal with missing or incomplete data, or data that is inconsistent with other data in the system (fault tolerance);
- The *EIS* has alternative implementations of essential processes that can be run manually. Additionally, data that normally enters the *EIS* in an automated way, can also be provided by hand. (fallback options);
- The *EIS* has ways to get back to normal mode when the large-scale disruption is over. This means that data that has been handled in manual mode or by atomic processes rather than by integrated processes can be merged with existing data to provide a consistent, but not necessarily complete, picture of the disrupted period (recoverability).

We consider *Context-Awareness* (*CA*) and *Risk Management* (*RM*) as key underlying paradigms in this regard. *CA* relates to the ability to sense that the *EIS* is operating out of bounds with respect to the set of acceptable values of the environmental parameters. The field of *RM* can provide practices for fault tolerance, fallback options, and recoverability.

As it concerns *CA*, Alferez and Pelechano [11] claim that it may be useful to translate the ideas of adaptation in the natural world to software, assuming that such adaptations are carried out in response to changing conditions in the surrounding physical environment and/or in the supporting computing infrastructure; this is referred to as *CA*, especially as far as *EIS* are concerned [11]. Even though the system would not be expected to

reconfigure itself for an unknown situation, it could at least sense that it is in such a situation. This is in line with the views of Dey et al., who already suggested in 2001 that context-aware *ICT* (*Information and Communication Technology*) applications should make use of the context that is relevant for the interaction with users; by "context" they mean information that concerns the state of people, places, and objects [12]. In further studies, Dey and Newberger argue that context information is typically gathered in an automated fashion [13, 14].

The *RM* field has traditionally dealt with making systems more robust against outside risks. This is being achieved by assessing potential risks on beforehand, classifying their likelihood and impact, and (based on the severity of the risk) providing mitigation measures to deal with the risk [16]. The *RM* field deals with disruptive events rather than with normal variability. *RM* thinking in general can be very useful for *EIS* robustness improvement. Several *RM* techniques such as *Failure Mode Effect and Criticality Analysis* (*FMECA*) and *Fault Tree Analysis* (*FTA*), are typically used for mechanical systems. We argue that *FMECA* and *FTA* could also be useful for the many interdependent components of an *EIS*.

This paper proposes innovative directions for more robust *EIS*, inspired by insights from *CA* and *RM*, while acknowledging the "emerging" nature of such research, characterized by insufficient existing experience on (and validation of) the feasibility of such "disaster-proof" *EIS*.

The remaining of this paper is structured as follows: A problem elaboration follows in Sect. 2. Related work analysis and corresponding conceptualization (featuring *CA* and *RM*) are presented in Sect. 3 and Sect. 4, respectively. Section 5 shows the possible application of *CA* and *RM* to disruptive events. Section 6 is proposing a corresponding method. Finally, Sect. 7 concludes the paper.

2 Problem Elaboration

This section elaborates the problem, by highlighting the difference between (i) normal variability and (ii) disruptive events:

(i) is about the regular variability in business processes and easy-to-predict situations, for example: "Supplies are delivered late and production processes need to be rescheduled based on the late supply", "Information provided by a business partner is incorrect and needs to be corrected", "There is shortage of a product and an alternative supplier needs to be selected to deliver the missing supply", "The agreed payment date is not met by a customer and a reminder needs to be sent", and so on. This is all well-manageable, by just assuming different possible situation variants and preparing (at design time) corresponding *EIS* variant actions. In these straightforward cases, situation awareness can be a solution: data indicates that the system is in an unwanted or inconsistent state (but a state that has been foreseen). The business logic in the *EIS* can choose the pre-defined rules to deal with the system state accordingly.

(ii) is about things that are not predictable at design time or where the likelihood of the occurrence of the event is considered to be so low that implementing variant actions

in the *EIS* is seen as too costly for something that may never happen. For example: one would not know at design time what disruptions in the business processes could be caused by an earthquake, a virus outbreak, or a military conflict.

In this work, we do not address (i) where there is already much knowledge and experience [17]. As it concerns (ii) however, we observe insufficient knowledge and lack of exhaustive experience. Hence, we explore the handling of disruptive events by *EIS* in the current paper.

In this regard, we firstly define the four essential (in our view) aspects of any *EIS*, namely: data, operation, quality of service, and public values – see Fig. 1.

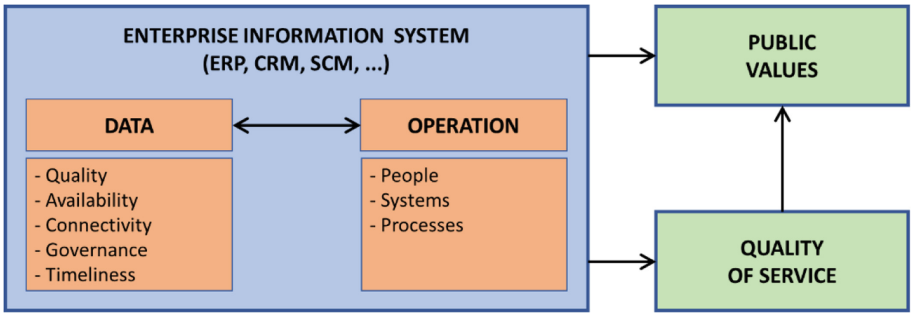

Fig. 1. The essential aspects of an *EIS*

As the figure suggests, there are several key issues that are fundamentally underlying the functioning of any *EIS*, namely: DATA (any *EIS* is about gathering, processing, and exchanging data, and for this reason the data availability, quality, timeliness, and so on are considered essential), OPERATION (one fundamental thing about an *EIS* is certainly its operation – what it does, how it does it, how different business processes are synchronized, and so on), QUALITY OF SERVICE (the quality of the services delivered by an *EIS* is the reason for its existence – going below a quality "threshold" would be considered as a failure), and PUBLIC VALUES (it is of crucial importance that in its operation, an *EIS* is not violating public values, such as privacy, accountability, and so on [31]). With these four aspects, we do not claim exhaustiveness and we only argue that they are essential for maintaining the overall value of an *EIS*. We briefly discuss each of the aspects below:

[*DATA*]. As suggested above and in line with [2], *EIS*' dealing with data is a matter of the timely availability of data, its quality, and the way it is transferred and governed. Any *EIS* can deal with data variability, such as data entry errors, formatting errors, brief connectivity interrupts, or late availability of data. Dealing with disruptive events is much harder. Think, for instance about: encrypted information that cannot be decrypted anymore (quality); a natural event wiping out a key supplier of data where the data is lost forever (availability); damaged Internet cables to islands leading to a disconnect of weeks or months (connectivity); a request to immediately provide information stored in

the *EIS* to the Police after a terrorist attack (governance); receiving an invoice more than a year late after the annual budget has closed (timeliness).

[*OPERATION*]. The correct functioning of an *EIS* assumes the availability of a certain number of people to operate it on a day-to-day basis [18], as well as availability of hard- and software and clear processes. Organizations have catered for normal variance such as people leaving the organization, for hardware crashes and software maintenance, and for people not always following procedures. Nevertheless, a disruptive event could cause the available operating staff to be reduced below a level where the system can still function (think, for instance about the effects of a virus pandemic, or a long-term strike of key personnel). Disruptions could also cause massive hardware or software unavailability, or breaching procedures on a large scale.

[*QUALITY OF SERVICE*]. Quality-of-service is key as it concerns the adequate functioning of an *EIS* for its external stakeholders [19]. Variability means that some circumstances may assume lower quality-of-service (e.g. delays) for a limited period of time, for example: during public holidays, financial IT services are unavailable. A disruptive event however could cause service quality deteriorations for a period not limited in terms of time (think, for instance about what a state of emergency could cause in a country, e.g. enforcing businesses to stop offering some services/products for an undisclosed time period). Another example would be events that cause a significant decrease in the quality-of-service: e.g., a cyber-attack causes an organization to provide SCM track-and-trace information by phone on a daily basis instead of automated and in real time.

[*PUBLIC VALUES*]. Public values, such as privacy and accountability, are always part of what is demanded from *EIS* [20, 21]. Variability means that even though most public values are addressed in a way considered to be widely accepted in Society (for example, respecting privacy), those same values may be considered differently in some situations, again stemming from a wide public consensus (for example, disclosing privacy-related details of a criminal). A disruptive event nevertheless may lead to a definitive violation of public values (think, for instance about the possibility that a government declares a state of emergency and enforces an organization to disclose personal data stored in its *EIS*).

We obviously cannot design an *EIS* for every situation that may occur, so the operation of the *EIS* is usually limited to situations that stay within certain bounds, which typically do not include the effects of disruptive events. We suggest to expand the functionality of an *EIS* (and therewith the organization) with the following three functions, so it can keep functioning in the case of a disruptive event:

- Firstly, we need to **detect that an anomaly has occurred**. Many *EIS* do not specifically define, measure and guard the acceptable boundaries for variables in the environment that allow it to function properly. This makes it impossible to automatically detect that the state of the environment is out-of-bounds. As we will see later, *CA* is of use here.
- Secondly, the *EIS* should **continue to function** as much as possible in spite of the state of the environment being anomalous. This asks for a *Risk-Based, Robust Design*

of the *EIS*, where critical parts can independently keep functioning when other parts of the system fail as a result of the event, and where inconsistent components can either be switched off, switched to manual operation, or by-passed.
- Thirdly, when the event is over, and the environment (slowly) returns to normal, the *EIS* would usually still have gaps in its data, internal inconsistencies, and procedure violations. *Resilience* therefore needs to be built-in to the *EIS* to allow the *EIS* to **return to its normal state** again.

As mentioned in the Introduction, we will identify opportunities and propose solution directions with regard to those challenges, inspired by studies touching upon CA and RM – this follows in Sects. 3 and 4, respectively.

3 Context-Awareness: Related Work and Conceptualization

This section covers related work, both from ourselves and from others, leading to a conceptualization for *CA*.

3.1 Analysis

We analyze firstly our previous work followed by relevant work of others.

As it concerns our previous work: In [22], we have analyzed different ways (in particular based on Bayesian Modeling and Semiotic Norms) of achieving application behavior adjustment, based on context data and assuming states that are foreseen at design time. Related to this, we have considered in [23] the application specification itself, making it explicit that following context changes, the application behavior is to be updated accordingly. In [2], we have taken a systemics [24] perspective for *CA*, addressing the environment and its changes, to which the system should adapt. In [4], we have considered three system adaptation perspectives with regard to context-aware systems, namely: (a) driven by the goal of optimizing the system-internal processes; (b) driven by the goal of maximizing the user-perceived effectiveness; (c) driven by the goal of achieving sensitivity to public values. Further, we have explicitly established that in each of those cases we have a different perspective of the context – as the context can relate to what is happening inside the system; to the user, or to public values. Nevertheless, we have only considered states that are foreseen at design time. Finally, in [25], we have studied business process modeling from the perspective of *CA*, addressing in particular business process variants – different business process variants could be relevant to corresponding context situations. Hence, our earlier research only relates to the "normal variability perspective" but not to the "disruptive events perspective".

As it concerns related work: Anind Dey is among the most recognized researchers addressing *CA* [12, 13]. He has improved our understanding of the notion of context and made serious progress in the development of context-aware applications. We argue nevertheless that he has not explicitly considered the challenge of tackling disruptive events, when system states cannot be foreseen at design time. The same holds for many R&D *CA* projects, such as AWARENESS [5]. Bosems and Van Sinderen have considered "context-aware computing" as the combination of sensor, reasoning, and other

technology that provides systems with real-time awareness [26] but the "reasoning" has not been explicitly addressed and is mainly related to Event-Condition-Action (ECA) rules [27]. In our view, ECA-rules are only limited to situations that are known at design time. The useful survey of Alegre et al. [28] is mainly focused on the development of context-aware applications as well as on the consideration of public values but not on the "disruptive events perspective". The same holds for the works of Alférez and Pelechano [11] – they consider the dynamic evolution of context-aware systems, the development itself, and the relation to web services, still not explicitly distinguishing between the "normal variability perspective" and the "disruptive events perspective". And the same holds for the service-orientation perspective as proposed by Abeywickrama [29].

Even though we do not claim exhaustiveness with regard to the related work analysis, we are convinced that it covers some of the most representative researchers and works relevant to the problem considered in this paper.

Hence, we argue that it is still an open question how to effectively extend context-aware systems, such that the "disruptive events perspective" is adequately covered.

For this reason, the conceptualization presented below (that is actually inspired by the works mentioned above) is only providing a general basis. It will subsequently be used in the following sections featuring proposed solution directions.

3.2 Conceptualization

Inspired by previous work [2, 4, 22], we essentially refer to concepts as presented in the meta-model for context-awareness (see Fig. 2 – left), which is built using the notations of the UML Class Diagram [30]:

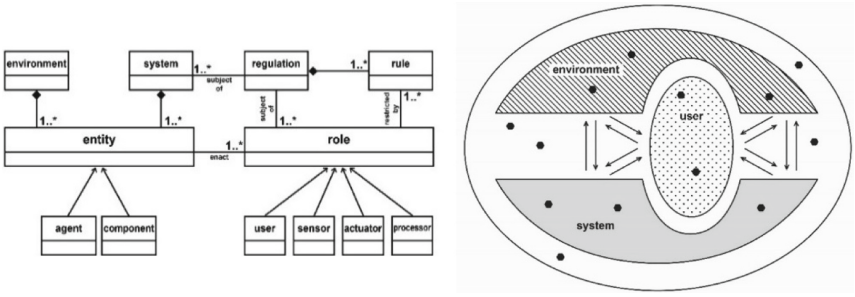

Fig. 2. Left: Considered meta-model for context-awareness (Source: [4], p. 197); Right: Considering the notions of *system*, *environment*, and *user* (Source: [2], p. 140)

Looking at the meta-model, we consider a *system* and its *environment*. Both are composed of numerous *entities* which in turn can be *components* (not pro-active) or *agents* (pro-active and intelligent). One *entity* (an *agent*, for example) can enact many different *roles* (and in the current paper, we limit ourselves to four role categories, namely: *user*, *sensor*, *actuator*, and *processor*) that are restricted by corresponding *rules* and are subject to *regulations*. A regulation in turn is composed of *rules* and is affecting not only the *roles* but also the *system* as a whole.

It is always a question whether we consider the *user* to belong to the *system* or to the *environment*. From one point of view, the *system* is driven by the goal of delivering something to the *user* and hence, the user is to be considered part of the *system*; nevertheless, from another point of view, the *user* is not among the entities that are delivering the product/service because the *user* is consuming it and hence the *user* is not to be considered part of the *system* (and is thus part of the *environment*) [2]. It is therefore not surprising that a lack of consensus is observed about how the *user* is to be considered. Hence, we clearly distinguish between: (i) what belongs to the *system*; (ii) what belongs to the *environment*; (iii) what belongs to the *user* (see Fig. 2 – right).

Further, in line with what was stated above: there are items that neither belong to the *system*, nor to the *environment*, nor to the *user*.

Finally, those "items" (visualized in Fig. 2 (right) as small black hexagons) actually reflect ENTITIES from the meta-model – see Fig. 2 (left) and they in turn fulfill actor-roles (ROLES, for short), for example: if a manager analyzes sales information, then (s)he is fulfilling the *role* "data analyst".

In summary, there is interaction among entities (fulfilling corresponding roles) in several perspectives: between system and environment; between system and user; between environment and user. Other entities are not involved in interactions, at least as it concerns the system under consideration.

For achieving *CA*, **sensors** that provide context information upfront are considered to be an instrumental enabler for adapting the system behavior. Bare sensor data is useless for this purpose. It has to be combined with rules for establishing the context state and changes in the context state. **Data analytics** can be used to further analyze the context state over time. Learning algorithms can provide us with expected behavior of the environment in the future, based on analyzing the trends from the past, and can, for example, predict the expected behavior of a stakeholder [7, 46]. Even though we are using historical data for this, we can use this data through learning and prediction algorithms for getting insight in what is most likely to happen in the future. Nevertheless, for the sake of brevity, we are only addressing sensor-driven CA in the current paper and do not elaborate on the data analytics techniques.

It is important to note that the current section is featuring the adaptation of the system behavior as it concerns *CA* Applications; this is driven by changes in the environment, in system-internal processes, and/or in processes that connect the system and its environment. But all those changes have been envisioned at design time; hence, when an unforeseen change would occur, not considered at design time, the system would be driven by algorithms that are nevertheless only considering expected changes. Thus, we argue that such a "prescribed" behavior would be of limited use in the case of a disruptive event – in such a situation, the environment would have changed to a "level" not expected at design time. For this reason, it is not surprising that in most *CA*-related literature, the perspective of an unexpected state of the system or the environment is missing.

4 Risk Management: Related Work and Conceptualization

For *RM,* we again cover the analysis first and then the conceptualization.

4.1 Analysis

Although there are many papers about information systems for *RM*, the number of papers on *RM* for *(E)IS* is comparatively small. González-Rojas and Ochoa-Venegas [32] show a decision model for the purchase or development and implementation of *EIS*, and indicate that most approaches do not consider risk attributes. Scott & Vessey [33] discuss risks in *EIS* implementation. Their risk factors focus mainly on the internal organizational risks for successful implementation. Their paper does, however, briefly mention the ability to withstand environmental change but only focuses on reactive measures from a management perspective to deal with external change. Broad [34] shows how a risk mindset can be an integral part of the systems development life cycle, and explicitly mentions risk assessment as a key ingredient for the development of information systems. The *RM* Framework (*RMF*) on which it is based, was developed by NIST [35]. The focus of the *RMF* is just on privacy and security, concerning the public values aspect in Fig. 1. O'Donnel [36] touches upon an important aspect of *RM*: the event identification phase, that links closely to the notion of *CA* in the previous section. This phase is one of the eight phases [37] from the Enterprise *RM* (*ERM*) field. ERM focuses on external events that can disrupt the enterprise's goals, but not particularly the *EIS*. In that sense, it is close to the ISO 31000 standard [38] that also focuses on enterprise risk rather than on external risks for the correct functioning of the *EIS* within the enterprise.

An *EIS* is a system. Therefore, another source of information for designing robustness into *EIS* is the systems engineering literature. Technical systems are designed in such a way that disruptive effects on the system are minimized. Still, not all methods for systems engineering explicitly address risk as part of the design methodology. The systems engineering sources that do, such as [39], still focus on risk analysis to study the potential failure of the system itself, rather than the system failing due to extreme events from the outside. The techniques that are discussed can, however, still be used to study external events. Important examples are Failure Mode Effects and Criticality Analysis (FMECA) as well as Fault Tree Analysis (FTA) [39]. The NASA Systems Engineering Handbook [40] looks at the design of mission critical systems, where the *RM* is further specified [41]. Here, continuous *RM* and risk-informed decision making are the basis of the design of complex systems. These sources, and the systems engineering sources in general, are focusing on how to design a system, and how to manage the design or construction of the system (project risk). Therefore, they mainly focus on *known risks* rather than on the *unknown threats* that we consider in the current paper.

4.2 Conceptualization

ISO 31000 [38] defines **risk** in a rather broad way: "the effect of uncertainty on objectives", where "risk is often expressed in terms of a combination of the consequences of an event and the associated likelihood of occurrence". This follows a broad set of literature that defines the *expected value* of the risk as *likelihood x consequence*, where

"likelihood" is the chance of the risk event occurring and "consequence" is some expression of an objective that can be hurt (e.g., safety, throughput, cost, customer satisfaction). Given the fact that the types of risks we are looking at all very-low-probability, very-high-impact risks (often termed *black swan risks* [42]), this is a fallacy. When the risk event fires (with a very low probability), we have to deal with the major consequence. When it doesn't, nothing happens. The expected value does not represent this in any useful way. Because many of these risks exist, it is on the one hand impossible to list all risks, while on the other hand some of these highly improbable risk events will actually fire. Since we are looking at *rare events* where the likelihood of occurrence of the event is extremely low, there is so little data available that the actual probability is often completely unknown, making it even harder to use the "likelihood x consequence" formula to decide on the relative importance of a risk. As the *Handbook of Systems Engineering and Management* [43] states (p. 180): "Risk of extreme events is misinterpreted when it is solely measured by the expected value of risk".

Therefore, we have to use a completely different approach. Instead of looking at the likelihood and consequence, we only look at the consequence. The fact that *something* could render our *EIS* useless for several months is what counts, not the calculation how often this would occur on average. We call this approach **Consequence-Based Risk Management** (*CBRM*). The *CBRM* approach stems from natural hazard research and climate change research where the most vulnerable locations are studied first. Taking this approach to *EIS*, we can define the following important terms:

- A **risk event** is an uncertain discrete occurrence that, if it occurs, would have a positive or negative effect on achievement of one or more objectives [44]. We focus mostly on the negative consequences in this paper.
- A **consequence** is the possible outcome of a risk if it occurs [44]. In our case, this relates to the intended functions of the *EIS* that need to be fulfilled. The consequence can be measured as the reduction in the agreed *EIS* service level.
- **Criticality** is defined as the importance of a component in the *EIS* to be able to fulfill the *EIS* functions.
- **Vulnerability** is defined as the (qualitatively or quantitatively assessed) likelihood that an *EIS* component will be exploited by the occurrence of a risk event.
- **Recoverability** is defined as the time it takes after the risk occurred to bring back the normal service level of the *EIS*.
- **Robustness** relates to decreasing the consequences for a wide set of risk events. This can, e.g., be done by decreasing the vulnerability of critical *EIS* components, by increasing the *EIS* recoverability or by decreasing the criticality of the involved components.
- **Enterprise *RM* (ERM)** is the integrated application of *RM* across the entire business, addressing all levels of risk, including strategic, business, corporate, reputation, portfolio, program, project, technical, safety, etc. [44].

Given the discussion about the inapplicability of the concept of expected value, and the fact we are dealing with rare events, the focus on any method trying to improve the robustness should start with identifying unwanted **consequences** for **critical components** in the *EIS*. In case there is information about the vulnerability of such components,

the most vulnerable component can be addressed first. Several methods to deal with vulnerable critical *EIS* components exist, as we will see in Sect. 5.2.

5 The Disruptive Events Focus

5.1 Context-Awareness

At the end of Sect. 3, we have concluded that most current *CA* solutions consider expected changes, which makes *CA* to be of limited use when it comes to disruptive events. In this section we therefore focus specifically on situations assuming unexpected changes and go beyond what is predicted/expected at design time. Several examples:

- A factory of a supplier is completely shut down (e.g., during a pandemic) but the *EIS* would only start finding this out and alert the organization when the first shipments would not arrive on time.
- A user is physically unable to connect to an *EIS* (e.g., during a disaster) but business processes of this user are fully dependent on the ability to use the *EIS*.
- Critical infrastructure is down for a long period (e.g., during an outage) and as a result of this, key *EIS* components are unable to operate, restricting the overall performance of the *EIS*.

As can be seen from the above examples, the types of disruption and their effects are different from normal variant situations. As a first measure in addressing such cases, we could aim at **sensing** what is going on, where the *EIS* is able to sense the occurrence of a disruptive event. In this regard, we count on KPI (Key Performance Indicators), which means that the designer should be able to identify outlier situations in each of the key modules of the *EIS*. Here, sensors and data feeds that are normally not considered for the day-to-day operation of the *EIS* would play an important role.

As a second measure, we could look into the **problem localization**. The *EIS* itself should always be capable of establishing which of its key modules is down or in an inconsistent state. This measure certainly relates to the first one as it also involves sensing, but this time for the *internal* state.

As a third measure, we could look into ways to **bypass the inoperative module**. This is a serious design challenge assuming "emergency *EIS* behaviors" – the designer should have established all possible scenarios featuring *EIS* "running with less engines". For example, if an *EIS* has 4 key modules: M1, M2, M3, and M4, then it should be known whether it can run without one of M1, M2, M3 or M4 (or even without a combination of several of the modules), and if yes – how. Then, if in the case of a disruptive event a problem localization has been established, the affected module is to be excluded from the system, where other modules are informed to ignore or bypass this inconsistent or faulty one. This would allow the damage to remain mainly local while many other (essential) system functions would stay available.

As a fourth measure, we consider **recoverability** – the *EIS'* ability to re-establish normal functioning, after the inoperative module(s) go back to normal.

5.2 Risk Management

From a *RM* perspective, several measures can help to deal with large-scale, unexpected changes. We provide three possible measures below:

As a first measure, we advise to make the *EIS* component **less vulnerable**. Example strategies for the data and operation aspects of Fig. 1 are to store data in human readable formats so humans can take over some processing functions if the *EIS* does not operate anymore. AI (Artificial Intelligence) techniques and business process automation can be used in case the operators are unavailable and the system needs to continue operating. In a sense, components become less vulnerable if fallback options for the operation of the component exist (note that the criticality of the component did not change: components are still critical for obtaining the agreed *EIS* service level).

As a second measure, we advise to make the *EIS* components **less critical**. Example strategies for the data and operation aspects of Fig. 1 are to duplicate data to multiple locations for the data aspect, and to have sufficient extra staff in multiple locations - for the operations aspect. Reliability analysis teaches us that criticality of a components goes down if we **duplicate** that component [39]. If one has many copies of something, it does not matter so much if one of the copies gets lost. *Blockchain* [45], for instance, explicitly uses duplication of records over so many servers that integrity of the data can be guaranteed because the system continuously checks if all copies are the same, and it is almost impossible to change the value in all servers at once to make an undetected change to the data.

As a third measure, we advise to make the *EIS* components **more resilient**. One of the biggest problems is starting up an information system that has been in an unplanned state. Internal data is inconsistent due to the risk event happening and starting up is hampered by the system wanting to maintain consistency all the time. Under normal variability, the consistency checks are a good thing: they guarantee that every time a small variation occurs, it does not go undetected and corrective actions can be performed. When there are thousands or millions of inconsistencies, this is not an option. Rather than trying to stubbornly maintain consistency, the system should be able to move to a more lenient state where most inconsistencies are tolerated (but can still be reported). This helps the system to go back to a working state after the disruption has passed, and thereby it makes the *EIS* more resilient, since resilience was defined as the ability for the *EIS* to return to its normal state again.

6 Application of Context-Awareness and CBRM for Robust and Resilient Enterprise Information Systems

There are several steps in designing a robust *EIS* that roughly follow the steps in any *RM* method [34, 35, 41, 43, 44]. We adapted the *RM* method by starting with the consequences of the risk event rather than the causes or risk events themselves. *CA* is used in several of the phases to enable *CBRM*.

Phase 1. Risk Identification and Analysis

Step 1. Identify the objectives of the EIS. The **Quality of Service** and the **Public Values** aspects at the right of Fig. 1 are the starting points for defining the objectives of the

EIS. As long as the system stays within the defined levels of service and adheres to the defined public values such as safety and privacy, the system is operating normally. These objectives should be identified and related to the **components** of the system. Which component or chain of components in the operations or data aspect of Fig. 1 is directly or indirectly responsible for reaching each business objective? Of course, the objectives might be taken from a design document of the *EIS*, but there could be a difference between the intended use of the *EIS* at design time, and the actual use when carrying out the CBRM study.

Step 2. Identify the critical components of the EIS. Based on the analysis of step 1, we can see which components are responsible for fulfilling many of the objectives, and which components are responsible for fewer objectives or even just one objective. Combined with a **usage analysis** (how often is each function of the system used) and a **business analysis** (how much money, long-term customer relations, or corporate responsibility is involved in the (in)ability to use such a component), the components of the EIS can be ranked in terms of their criticality.

Step 3. Identify the vulnerability of the components of the EIS. For some of the components, it may be known that vulnerabilities exist. A *worst-case analysis*, for instance, can be based on a long list of consequences consisting of natural disasters (e.g., pandemic, flooding, earthquake) and man-made disasters (e.g., regional war, cyber-attack, ransomware) on the system. Rather than trying to study the likelihood of each event, it is sufficient to see which components of the system would be impacted by such events. The result of this step is a list of **components** that can be ranked from components showing up often in the analyses (**vulnerable**) to components showing up less often (not so vulnerable).

Step 4. Combine criticality and vulnerability. Our priority should be based on those components that are critical and vulnerable. From this analysis, an **overall ranking** can be made to show which components decrease the **robustness** of the system most. Personal preferences of the IT managers and overall enterprise management can also be taken into account into the selection of the most vulnerable components. Qualitative assessment and experience can be an important addition to the priority ranking shown above, since it may be hard to take into account non-quantifiable factors in the priority ranking. From a CBRM perspective, we have now identified those components for which the consequences of risks would be most devastating on being able to maintain the agreed service level of the *EIS*.

Phase 2. Mitigation Planning

For each of the identified critical and vulnerable components, a risk mitigation strategy should be designed, in line with *RM* practices. This mitigation strategy can be one of [44]:

Strategy 1: **Accept** the risk. It is possible that the organization is not able to adapt the component that is critical and vulnerable. Still, there is now awareness that this component is a risk for the enterprise, and monitoring can be put in place to sense that the conditions in which the component can function are violated (see Phase 3).

Strategy 2: **Reduce** the risk. One of the strategies from Sect. 5.2 can be used to make the component less vulnerable, less critical, or more resilient. Often this involves sensing as well, e.g., to see when a fallback option needs to be switched on (see Phase 3).

Other Strategies: Risks can also be Transferred (e.g., insured) or Avoided (e.g., by removing vulnerable components) [44]. These strategies can be applied but since they do not consider *CBRM* and *CA*, they are not covered further in this paper.

Phase 3. Monitoring

Although one might think that it is possible to make the system totally fail-safe by reducing all risks to zero, in practice this is undoable due to time and budget constraints. Therefore, many of the solutions will be implemented partially, where an effort can be made to switch the system manually to another state when needed, or to reduce the risk only when the consequence actually occurs. A strategy where repairs are not immediately made but we make a plan for dealing with the consequence when it happens, is called a **contingency strategy**. The "Accept" strategy from Phase 2 above is an example of using contingency. In order to be able to apply such a strategy, constant monitoring of the **context** of the system needs to take place to assess that the system gets into a state that is conflicting with the assumptions of the critical modules of the system. It is important for *CA* to distinguish between normal variability and consequences of disruptive events. Thresholds for different contextual variables need to be set for the *CA* algorithms, based on the boundaries within which the critical modules function correctly.

Note that one of the requirements of the above method is that the *EIS* consists of **components**, each with a clear **function**, that can be distinguished and for which the interfaces are known. When the *EIS* is monolithic, the above strategies cannot work, and neither the provides solutions from CBRM, nor the provided insights from CA will have added value.

7 Conclusions

Even though most current Enterprise Information Systems (*EIS*) appear to be adequately dealing with variability, they would often fail when affected by a (large-scale) disruptive event. As we have studied in the current paper, Context-Awareness (*CA*) is a useful paradigm as it concerns the capturing of changes occurring in the *EIS* environment, including changes that lead to unplanned states. *CA* can be accomplished by adding an extra function for monitoring the environment, aiming at establishing whether or not the *EIS* needs to adapt accordingly. As mentioned above, we have been inspired by the strengths of this paradigm as it concerns variability, and we propose extending the use of *CA* towards disruptive events, by considering three measures, namely the capturing/sensing of an unexpected environmental state, the localization of the problem (in terms of affected *EIS* modules), and bypassing (if possible) of inoperative modules. Further, we have studied the relevant strengths of Risk Management (*RM*), considering relevant techniques, such as FMECA and FTA, in the light of a particular approach, namely Consequence-Based Risk Management (CBRM). On that basis, we have proposed three corresponding measures, namely: to make *EIS* components less vulnerable, to make them less critical (e.g., by duplicating them), and to increase their resilience.

Finally, we have proposed an integrated method featuring the application of *CA* and CBRM for robust *EIS*.

The combination of *RM* and *CA* to deal with disruptive events for *EIS* is a conceptual solution. For validation of its usefulness and applicability, it has to be tested in either a simulation or a real application. Disruptive events luckily do not happen every day. Further research will therefore focus on testing the method on a simulated *ERP*, *SCM* or *CRM*, and seeing whether the *RM* methods can be applied and whether the *CA* functions can be automated to flag the disruptive events correctly, and trigger the right corrective action that was defined as a result of applying consequence-based risk management and reliability engineering in the *EIS*.

Future work will focus on elaborating the proposed CA and CBRM methods and on testing the approach on a simulated or real case.

References

1. Snoeck, M.: Enterprise Information Systems Engineering, the MERODE Approach. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10145-3
2. Shishkov, B.: Designing Enterprise Information Systems, Merging Enterprise Modeling and Software Specification. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-22441-7
3. Shishkov, B., van Sinderen, M., Verbraeck, A.: Towards flexible inter-enterprise collaboration: a supply chain perspective. In: Filipe, J., Cordeiro, J. (eds.) ICEIS 2009. LNBIP, vol. 24, pp. 513–527. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01347-8_43
4. Shishkov, B., Larsen, J.B., Warnier, M., Janssen, M.: Three categories of context-aware systems. In: Shishkov, B. (ed.) BMSD 2018. LNBIP, vol. 319, pp. 185–202. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94214-8_12
5. Wegdam, M.: AWARENESS: a project on context AWARE mobile NEtworks and ServiceS. In: Proceedings of 14th Mobile and Wireless Communications Summit. EURASIP (2005)
6. Kopják, J., Sebestyén, G.: Comparison of data collecting methods in wireless mesh sensor networks. In: IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Kosice and Her-lany, Slovakia (2018)
7. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 3rd edn. Morgan Kaufmann Publ. Inc., San Francisco (2011)
8. Reuters: Italian PM Orders Businesses to Close All Operations. In: The Guardian - International Edition, London (2020)
9. Takizawa, K.: Resilience of communities affected by the great east japan earthquake and restoration of their local festivals. In: Bouterey, S., Marceau, L. (eds.) Crisis and Disaster in Japan and New Zealand. Palgrave Macmillan, Singapore (2019)
10. Shibata, Y.: Writing Shanghai, the atomic bomb, and incest: homelessness and stigmatized womanhood of Hayashi Kyōko. In: Bouterey, S., Marceau, L. (eds.) Crisis and Disaster in Japan and New Zealand. Palgrave Macmillan, Singapore (2019)
11. Alférez, G.H., Pelechano, V.: Context-aware autonomous web services in software product lines. In: Proceedings of 15th International SPLC Conference. IEEE, CA, USA (2011)
12. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact. **16**(2), 97–166 (2001)
13. Dey, A.K., Newberger, A.: Support for context-aware intelligibility and control. In: Proceedings of SIGCHI Conference on Human Factors in Computing Systems. ACM, USA (2009)

14. Papadimitriou G.: Future Internet: The Cross-ETP (2011). http://www.future-internet.eu/fileadmin/documents/reports/Cross-ETPs_FI_Vision_Document_v1_0.pdf. Accessed December 2011
15. Choraś, M., Kozik, R.: Machine learning techniques applied to detect cyber attacks on web applications. Log. J. IGPL **23**(1), 45–56 (2015)
16. Hopkins, P.: Fundamentals of Risk Management - Understanding, Evaluating, and Implementing Effective Risk Management. IRM (2012)
17. La Rosa, M., Van Der Aalst, W.M.P., Dumas, M., Milani, F.P.: Business process variability modeling: a survey. ACM Comput. Surv. **50**(1), Article 2 (2017)
18. Dietz, J.L.G.: Enterprise Ontology, Theory and Methodology. Springer, Heidelberg (2006)
19. Abeywickrama, D.B.: Context-aware services engineering for service-oriented architectures. In: Bouguettaya, A., Sheng, Q., Daniel, F. (eds.) Web Services Foundations. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-7518-7_12
20. Friedman, B., Hendry, D.G., Borning, A.: A survey of salue sensitive design methods. In: A Survey of Value Sensitive Design Methods, vol. 1. Now Foundations and Trends (2017)
21. Van den Hoven, J.: Value sensitive design and responsible innovation. In: Owen, R., Bessant, J., Heintz, M. (eds.) Responsible Innovation: Managing the Responsible Emergence of Science and Innovation in Society. Wiley, Hoboken (2013)
22. Shishkov, B.: Tuning the Behavior of context-aware applications. In: Shishkov, B. (ed.) BMSD 2019. LNBIP, vol. 356, pp. 134–152. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24854-3_9
23. Shishkov, B., van Sinderen, M.: From user context states to context-aware applications. In: Filipe, J., Cordeiro, J., Cardoso, J. (eds.) ICEIS 2007. LNBIP, vol. 12, pp. 225–239. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88710-2_18
24. Bunge, M.A.: Treatise on Basic Philosophy. A World of Systems, vol. 4. D. Reidel Publishing Company, Dordrecht (1979)
25. Shishkov, B., Mendling, J.: Business process variability and public values. In: Shishkov, B. (ed.) BMSD 2018. LNBIP, vol. 319, pp. 401–411. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94214-8_31
26. Bosems, S., van Sinderen, M.: Models in the design of context-aware well-being applications. In: Meersman, R., et al. (eds.) OTM 2014. LNCS, vol. 8842, pp. 37–42. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45550-0_6
27. Cano, J., Delaval, G., Rutten, E.: Coordination of ECA rules by verification and control. In: Kühn, E., Pugliese, R. (eds.) COORDINATION 2014. LNCS, vol. 8459, pp. 33–48. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43376-8_3
28. Alegre, U., Augusto, J.C., Clark, T.: Engineering context-aware systems and applications. J. Syst. Softw. **117**(1), 55–83 (2016)
29. Abeywickrama, D.B., Ramakrishnan, S.: Context-aware services engineering: models, transformations, and verification. ACM Trans. Internet Technol. J. **11**(3), Article 1 (2012)
30. UML: The website of the Unified Modeling Language (2020). http://www.uml.org
31. Shishkov, B., Janssen, M.: Enforcing context-awareness and privacy-by-design in the specification of information systems. In: Shishkov, B. (ed.) BMSD 2017. LNBIP, vol. 309, pp. 87–111. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78428-1_5
32. González-Rojas, O., Ochoa-Venegas, L.: A decision model and system for planning and adapting the configuration of enterprise information systems. Comput. Ind. **92–93**, 161–177 (2017)
33. Scott, J.E., Vessey, I.: Managing risks in enterprise systems implementations. Commun. ACM **45**(4), 74–81 (2002)
34. Broad, J.: Risk Management Framework. A Lab-Based Approach to Securing Information Systems. Elsevier, Amsterdam (2013)

35. NIST: NIST Special Publication (SP) 800-37 Revision 2, Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy. NIST, December 2018. https://csrc.nist.gov/publications/detail/sp/800-37/rev-2/final
36. O'Donnel, E.: Enterprise risk management: a systems-thinking framework for the event identification phase. Int. J. Acc. Inf. Syst. **6**, 177–195 (2005)
37. COSO: Enterprise Risk Management - Integrating with Strategy and Performance. Committee of Sponsoring Organization of the Treadway Committee (COSO) (2017)
38. ISO: ISO 31000 - Risk Management. International Organization for Standardization (ISO), Geneva (2018)
39. Blanchard, B.S., Fabrycky, W.J.: Systems Engineering and Analysis, 4th edn. Prentice-Hall, Upper Saddle River (2006)
40. NASA: NASA SP-2016-6105 Rev2: NASA Systems Engineering Handbook. NASA (2016). https://www.nasa.gov/connect/ebooks/nasa-systems-engineering-handbook
41. NASA: NASA/SP-2011-3422: NASA Risk Management Handbook (2011). https://sma.nasa.gov/sma-disciplines/risk-management
42. Taleb, N.N.: The Black Swan - The Impact of the Highly Improbable, 2nd edn. Random House, New York (2010)
43. Haimes, Y.Y.: Chapter 3: risk management. In: Sage, A.P., Rouse, W.B. (eds.) Handbook of Systems Engineering and Management, 2nd edn, pp. 155–204. Wiley, Hoboken (2009)
44. Hillson, D., Simon, P.: Practical Project Risk Management: The ATOM Methodology, 2nd edn. Management Concepts Press, Tysons Corner (2012)
45. Underwood, S.: Blockchain Beyond Bitcoin. Commun. ACM **59**, 15–17 (2016)
46. Borissova, D., Cvetkova, P., Garvanov, I., Garvanova, M.: A framework of business intelligence system for decision making in efficiency management. In: Saeed, K., Dvorský, J. (eds.) CISIM 2020. LNCS, vol. 12133, pp. 111–121. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-47679-3_10