An Integrated Dependability Analysis and Design Method for Distributed Systems Engineering

Claus Pahl
Free University of Bozen-Bolzano, Bolzano, Italy

Keywords: Dependability Engineering, Reliability, Availability, Safety, Security, Requirements Analysis, Design, Architecture, Software Quality.

Abstract: Dependability Engineering is a critical importance for the now huge number of software systems that are directly linked to people (and can damage them – be that physically or through security and data protection issues) and that operate in open and distributed environments that expose them to reliability problems. We present here an integrated dependability analysis and design methods for reliability, safety and security that targets beginners and that can be applied without specific tools support. We introduce this analysis and design method here in a concrete educational setting that is made up of beginners and works without tools. We will look at this specifically in the context of Internet-of-Things data with realistic application cases.

1 INTRODUCTION

Dependability quantifies several dimensions to predict how well and how long a system will operate. These dimensions typically safety, security, reliability and availability (ISO/IEC, 2011). Environments such as the Internet-of-Things (IoT), edge and cloud systems are particularly reliant on dependability.

While these aspects have been investigated, a comprehensive method that provides, firstly, an integrated view across all concerns, secondly, is based on a joined architecture perspective, and, thirdly, uses a uniform mechanism for the dependability analysis despite differences in the metrics and assessments being involved. We introduce here an analysis and design method for dependable systems in open distributed systems such as cloud, edge and IoT environments. Furthermore, we target a beginners setting not supported by specific tools to allow the methods to be brought to non-experts. We report this here in a teaching context, presenting our experience for a Master-level course on dependable systems analysis and design. This setting demonstrates the need for a uniform methods that can be used within the constraints of graduate computer scientists without deep experience and specific tools.

Reliability is one of the central dependability properties, largely capturing that a system performs as requested. Reliability starts with a non-functional reliability requirements analysis that then extracts functional reliability requirements and a design build on suitable reliability architecture patterns. Safety address the impact of a system onto its environment in terms of damages that can be caused. A hazard/accident analysis is the first step, followed by a corresponding risk analysis and a functional requirements and architecture steps. Security is concerned with the impact of the environments on the system, be that malicious or unintentional intrusions and their negative consequences. Again, a non-functional requirements analysis is following by a functional perspective and architectural design.

We present a method for dependable systems analysis and design for the three selected aspects. This has been evaluated for a number of application systems in the IoT and edge space. Key properties are the use of a table format to have a uniform structure and representation approach that does not rely on specific tools and also a joint architecture approach based on common architecture patterns.

The methods takes into account the needs of a non-expert setting. Here, the table-based analysis approach allows a uniform approach that is easy to convey. Also, architecture patterns as a common approach to address quality in software architectures is a key ingredient. Thus, the aim is not to support fully safety- or security-critical application development, but to create a wider awareness of dependability needs for less critical applications.

The remainder is organised as follows. Section 2

Pahl, C.
An Integrated Dependability Analysis and Design Method for Distributed Systems Engineering.
DOI: 10.5220/0010433102310238
In *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2021)*, pages 231-238
ISBN: 978-989-758-508-1

reviews the background on dependability concerns. In Section 3, we review the state-of the-art, both from an educational and technical perspective. The analysis and design method is introduced in Section 4 and the subsequently evaluated in Section 5. We conclude in Section 6.

2 BACKGROUND

Reliability, safety and security are well explored concepts (Mellor, 1992; Avizienis et al., 2004; Al-Kuwaiti et al., 2009; Walter and Suri, 2003), which is partly reflected in standards.

- Security: ISO 27005:2018 (ISO/IEC, 2018) focuses on security requirements.
- Reliability: IEEE Standard 1044-1993 (IEEE, 2010) is the IEEE Classification for Software Anomalies that covers the reliability concepts fault, error and failure.

In order to illustrate, but also to validate the proposed solution, we refer to the following use cases:

- Road Mobility and Automotive: here we consider connected cooperative autonomous mobility (CCAM), specifically focusing in examples on cars and other vehicles in a motorway setting (Barzegar et al., 2020b; Barzegar et al., 2020a; Gand et al., 2020; Le et al., 2019). This a technical setting that involves IoT infrastructure through sensors and actuators in cars and road-side, but also edge capabilities for local car coordination as well as remote cloud processing. Since cars with passengers are involved, reliability, safety and security are of highest importance.
- Industry 4.0: automated production scenarios equally require local coordination and processing, e.g., for sensor-guided robots in production lines, but also remote centralised processing of larger volumes of data in clouds (von Leon et al., 2019; Scolati et al., 2019; El Ioini and Pahl, 2018). As business-critical activities, reliability and security are important, as is safety of the operators and machines in the production process.

3 RELATED WORK

The importance of dependability and the need to consider this in non-expert settings has been recognised (Schoitsch and Skavhaug, ; Pahl et al., 2019). In this paper, however, only a broad strategic perspective is given. A concrete approach is lacking. In the wider context of distributed systems, approach to teaching and learning exist (Pop and Cristea, 2019). Here, the need to cover IoT, cloud and edge scenarios through architecture and also operations management is highlighted. Some works, such as (Michael et al., 2019), go deeper on specific technologies. Here, model checking is proposed a specific tool. However, we are looking at a broader analysis and design process coverage.

There is somewhat more technical progress. Some attempts have recently tried to integrate dependability concepts. (Shan et al.,) survey the current literature on all dependability aspects. In (Verma et al., 2019), the authors address a combined safety and security perspective. (Dobaj et al., 2019), specifically look at risk assessment in these two aspects.

In general, the need for a joint treatment is recognised (Serpanos, 2019), but not may integrated approach exist. The need for an integrated perspective is recognised for specific domains such as the automotive and mobility domain that we also focus on (Much, 2016).

This is where we aim to present an integrated approach that (i) integrates reliability, safety and security, but also (ii) requirements analysis with architectural design. However, it should be understood that our aim is to provide a method that covers all major aspects, but might not include deeper details that would be required for a professional industrial setting (but are not feasible in an educational setting).

4 DEPENDABILITY FRAMEWORK

The presentation of the dependability analysis and design framework is presented in two parts: firstly, a conceptual summary of the quality concerns, metric and their meaning, followed by, secondly, a process for analysis and design based on the common table structure.

4.1 Conceptual Framework

For each of the three aspects reliability (including availability), safety and security, we now present a conceptual map defining the key concepts and the relevant metrics.

4.1.1 Reliability

A basic conceptual map of the reliability aspect is presented in Figure 1 that follows (IEEE, 2010).

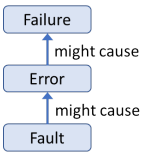

Figure 1: Reliability - Conceptual Map.

Often, availability, which in itself is an important concerns, is captured as part of the reliability concern. We reflect this and cover availability as part of the metrics:

$$
\begin{aligned}
AVAIL &= 1 - \frac{downtime}{uptime} \\
POFOD &= \frac{number\ of\ failures}{number\ of\ requests} \\
ROCOF &= \frac{number\ of\ failures}{total - elapsedtime} \\
MTTF &= \frac{1}{ROCOF} = \frac{total - elapsedtime}{number\ of\ failures}
\end{aligned}
\tag{1}
$$

Note that these are numerical dimensions. Later on in the security and safety aspects, we will also consider categorical aspects. These metrics need to be better explained in order to clarify the meaning and when they should be used. For instance, POFOD is suitable for rarely used functions such as emergency systems, while ROCOF is suitable for regulary and frequently used functions that are short in duration. For functions with a long duration, the MTTF metric is more suitable.

For availability, it needs to be noted that systems with the same availability might have different outage behaviours, see Figure 2. In many cases, many shorter outages (below some threshold) are less critical than one long one.

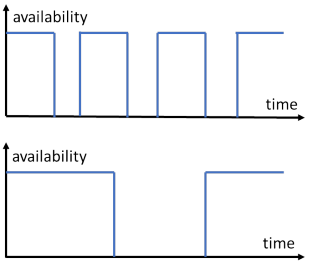

Figure 2: Graph Availability.

ROCOF captures the number of failures for frequently occurring events.

MTTF captures the likely time until the next failure, which should ideally not happen during the function execution of a long-lasting function.

These graphs help to clarify the semantics of the metrics. In addition, we provide information to the users on measurements and requirements values. At least in generic terms, we explained the difference between orders of magnitude, e.g., between 0.9 and 0.99 for availability and how those translate into time periods (such as a day or a year).

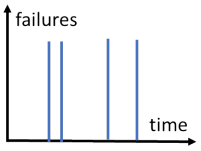

Figure 3: Graph ROCOF.

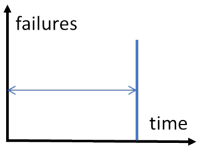

Figure 4: Graph MTTF.

4.1.2 Safety

We start again with a conceptual map, shown in Figure 5. The central concepts are hazards, which might turn into accidents. Accidents if they happen cause damage. The important task is to assess the risk that a hazard becomes an actual accident. The risk is defined through the severity and probability of the accident.

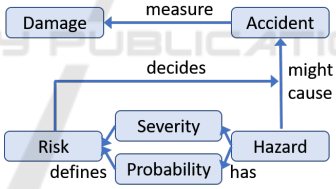

Figure 5: Safety - Conceptual Map.

In order to assess the safety criticality, the following metrics severity level, probability level and risk level with their respective value ranges are proposed

- Risk: tolerable, ALARP (as low as reasonably possible), intolerable)
- Severity: low, medium, high
- Probability: low, medium, high

Rather than being numerical, we assess these as a classified metric. The concrete values are common ones, but could be altered if needed.

4.1.3 Security

The conceptual map for security reflects threats and how they could be exploited by an intruder, as shown

in Figure 8. Starting with the assets to be protected, the individual security concerns such a exposure, threat and vulnerability can causally linked. A vulnerability can then be exploited by an attacker, but countered with a control mechanism.

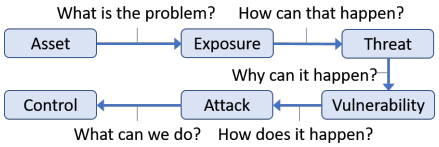

Figure 6: Security - Conceptual Map.

The relevant metric is here the security level, which like in the safety example uses classified, ordered labels: low, medium, high.

4.1.4 Discussion

While for reliability we have used quantitative metrics, for security and safety quantitative metrics are proposed. Reliability metrics are generally easy to monitor and measure. The others however are influenced by a multitude of factors that are more difficult to quantify. For instance, security is the summary of aspects such as integrity and confidentiality. Thus, labels as broader categories have been suggested.

4.2 Analysis and Design Process

4.2.1 Non-functional Requirements Analysis

All three dependability aspects start with a non-functional requirements perspective.

For **Reliability**, we propose a 2-step approach consisting of a fault-failure analysis and a mapping to corresponding metrics. The *Fault-Failure Analysis* associates faults and possibly resulting failures to each component:

Component | *Fault* | *Failure*

A concrete example of this for the automotive road mobility use case is presented in Table 1. The example focuses on hardware components as the more likely sources of failures.

Table 1: Fault-Failure Analysis.

Component	*Fault*	*Failure*
sensor	broken	no data
camera	dust	incorrect data
5G antenna	broker	no data up/down
GPS	no signal	no position

The *Dependability Mapping* associates metrics and their concrete values to the components and the relevant dependability concern.

Component | *Dep Concern* | *Metric* | *Value*

In Table 2, the values reflect assumed realistic requirements.

Table 2: Reliability Analysis.

Component	*Dep Concern*	*Metric*	*Value*
sensor	reliability	AVAIL	0.99
camera	reliability	POFOD	0.9
5G antenna	reliability	AVAIL	0.999
5G antenna	reliability	ROCOF	0.9999
GPS	reliability	AVAIL	0.999

For the next concern, **Safety**, we propose a 3-step approach consisting of a hazard-accident analysis, a risk analysis and a fault-tree based root cause analysis. The *Hazard-Accident Analysis* identifies the relevant hazards and possibly resulting accident. A concrete application of the analysis is given in Table 3.

Component | *Hazard* | *Accident*

Table 3: Safety Analysis.

Component	*Hazard*	*Accident*
external	object on road	crash (car/people)
antenna	fail to communicate	crash/system stops
edge	overload	car crash
external	security attack	car crash

The seconds step is the *Risk Analysis*, which associates accident probability and severity, from which an overall risk can be derived. A concrete example is given in Table 4.

Hazard | *Probability* | *Severity* | *Risk*

Table 4: Safety Risk Analysis.

Hazard	*Probability*	*Severity*	*Risk*
animal	low	high	ALARP
truck	medium	high	intolerable
antenna	high	high	intolerable

The third step is the *Fault Tree*, which is a *Root Cause Analysis*, presented in Figure 7.

The final dependability concern is **Security**. A concrete example is shown in Figure 5. This starts with a *Security Risk analysis*:

Asset | *Exposure* | *Threat* | *Vulnerability* | *Attack* | *Control*

Then, a *Security Policy* needs to be specified, for which a concrete example is given in Table 6.

Asset | *Level of Protection* | *responsibilities* | *Procedures and Techniques*

The analysis and design process shall now be summarised, see Fig. 1

Table 5: Security Risk Analysis.

Asset	*Exposure*	*Threat*	*Vulnerability*	*Attack*	*Control*
DB	data loss (integrity)	accidental deletion OR unauthorised user	no proper access control	unauthorised user accesses DB	authentication software
sensor	data manipulation	unauthorised use	no encryption	packet spoofing and manipulation	encryption
sensor	data manipulation	noise	redundancy	accidental	encryption, access control
sensor	data leakage	unauthorised use	no encryption	packet spoofing and manipulation	encryption

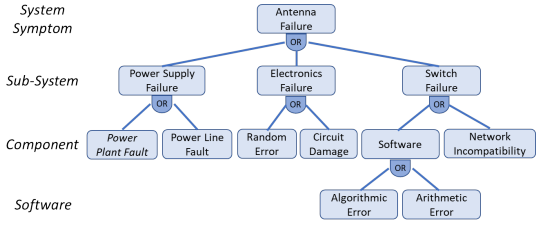

Figure 7: Safety - Fault Tree.

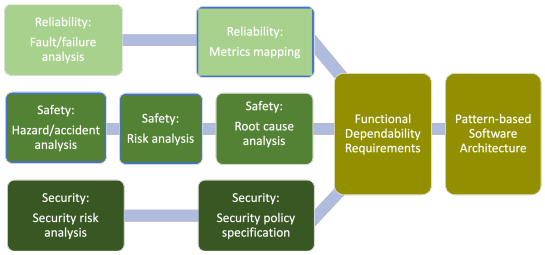

Figure 8: Analysis and Design Process.

Table 6: Security Policy.

Asset	*Level of Protection*	*Responsibilities*	*Procedures and Techniques*
car position data	high	in transit: communication infrastructure	encryption
DB / edge cloud	high	DB manager	access control and redundancy

4.2.2 Functional Dependability Requirements and Architecture

Now, we change the perspective from non-functional dependability requirements to functional dependability requirements, i.e., requirements for system components that help to prevent, detect or remedy (heal) the dependability requirements identified. The general objectives are avoidance, identification, remediation. We outline here some sample architectural strategies. In the context of this dependability design, we need an architectural style. We specify the key principles and patterns of this architectural style as follows: (i) Principles: redundancy, diversity; (ii) Patterns: MAPE-K, Protection System, Multichannel Architecture. Figure 9 illustrates the MAPE-K pattern. In the MAPE-K, K represents knowledge, for example rules of the form `if` $T > 20$ `then` *CloseValve* `else` *OpenValve* for T of type temperature and $T = 20$ for a self-adaptive heating system. Figure 10 shows the protection system pattern. Figure 11 shows the multi-channel architecture, which proposes to carry out the same task using different mechanism, and let a selector decide on the correctness of the result.

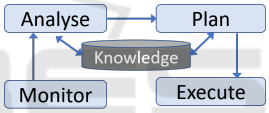

Figure 9: Layered ML Quality Management Architecture.

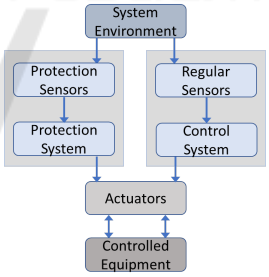

Figure 10: Protection System.

The patterns allow to select a concrete architecture that realises the principles of redundancy and diversity (Barrett et al., 2006; Pahl et al., 2018), which are known to aid dependability engineering. In order to avoid, identify or remedy dependability risks, these patterns can be utilised. For **Reliability**, the following components could be suitable:

- Reliability: a multichannel architecture to increase reliability.
- Availability: MAPE-K with with analysis and planning components for instance to utilise HADR (high availability and desaster recovery)

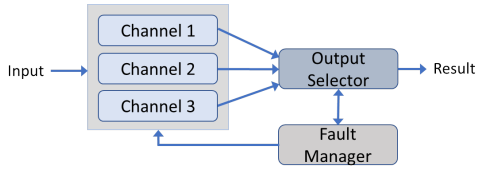

Figure 11: Multi Channel Architecture.

functions as actions.

For **Safety**, the following architectural components could be suitable: MAPE-K with monitoring to measure the relevant safety concerns such whether objects in the vicinity are still intact and react with an emergency shutdown in an accident situation (or even as a prevention measure before the actual accident happens). For **Security**, a protection system against external attacks is a solution.

We now illustrate this stage. Examples are presented in Table 7. The concerns of this architectural analysis and design step for security are as follows:

Layer of risk | *Control* | *architectural solution type*

Table 7: Architectural Design.

Layer of Risk	*Control*	*Architectural Solution Type*
technology / platform	login	protection
architecture / application	partitioning	distribution
asset / data / object / record	encryption	protection

Concrete functional components are firewall, encryption or access control.

5 EVALUATION

The methods targets a non-expert developer that might not have access to dedicated tools. A sample setting that matches these requirements is education. The presented method has been used over 4 years in the course *Requirements and Design for Dependable Systems* as part of an M.Sc. in Software Engineering where participants had a least a degree in computing, partly with industrial development experience. During this period, around 15 students in average have participated. Since this course is also catering for part-time students, the proportion of students with industrial experience, either prior to enrolling or concurrently experiencing, is more than 50 %.

For this context, the key evaluation criteria we have chosen are: (i) relevance of the method, i.e., how realistic and fit for purpose the proposed method is in industrial practice, (ii) non-expert suitability, i.e., how well the method is suitable for untrained developers without dedicated tools.

5.1 Relevance

The relevance criterion assesses the technical adequacy of the method, i.e., whether it reflects industrial practice and aligns with common conceptualisations. For the latter point, we already presented the alignment with standards regarding the conceptual scope in the Background section earlier on. A more detailed investigation of the criteria shows:

- Concepts: validity (complete and necessary) is given, 1) based on using standards and referring to relevant literature on dependability concepts, and, 2) is verified in concrete experiments (application scenarios) that we document below.
- Process: effectiveness (i.e., is usable and achieves the goals) has been demonstrated experimentally by applying the method over 4 years in 4 different application settings by more than 25 teams.

A number of domains have been selected, that vary in terms of the distribution of the setting, the number and form of people being involved, and the type of cyber-physical system in terms of hardware/embedded system components utilised. This is summarised in Table 8.

Table 8: Application Scenarios in IoT Settings.

Application Domain	*Properties*	*Source*
Road mobility	Safety, Reliability	EU H2020 Project 5G-CARMEN
Industry 4.0	Reliability	Microtec Scanning Devices – Wood Production
Health	Safety	Insulin Pump Case Study
Tourism	Safety	Dolomites Skiing

5.2 Suitability

While the previous section looked at the technical adequacy of the method (in terms of relevance for industrial practice), we now consider the suitability for the non-expert setting. The requirements here were:

- Comprehensibility, i.e., the the method is easy to learn within a given time frame (e.g., that of the course in question).
- Tool support, i.e., the feasibility of applying the method in a constrained setting without dedicated analysis tools.

For comprehensibility, we created a uniform table-based format for all dependability aspects that refers to metrics and measurements of common types for the analysis part. Equally, for the architecture part, the starting point were architecture patterns, which as industry practice are commonly used in software engineering and software architecture teaching. Thus, basing the method on common structures and mechanism aided the learnability.

For tool support, here no specific software tools were requires as only tables and block diagrams had to be created. Here, the simple presentation allowed open discussions and the use of blackboard in the validation setting of classroom-based teaching.

In terms of acceptance and usability of the method, from the student side, the high topicality of the problem context as well as the chosen application domains were appreciated. The method itself with is presentation elements and its proposed process has not caused problems. The difficulty here was more in understanding domain knowledge.

6 CONCLUSIONS

The dependability of modern software systems is due to their deep involvement not only in industrial production or organisational administration, but also in our everyday life of critical importance. Consequently this needs to be taught to students as well. For the classroom, we need a dependability engineering that takes on-board the critical concerns reliability, safety and security, but also does so in in a format suitable for the constraints of teaching. We use a table-based structure for analysis and integrated important metrics into it. The architectural design is based on architectural patterns to steer the system design towards important quality criteria.

Overall, the method does not aim at support fully safety or security-critical applications, but to provide improved dependability and in particular an improved awareness of the concerns for a wider range of applications. This is of particular importance for the described training context. Here, the uniformity of the modelling means should aid online presentation needs (Kenny and Pahl, 2005; Pahl et al., 2004; Murray et al., 2003; Lei et al., 2003; Melia and Pahl, 2009; Fronza et al., 2019). We have taken a first step towards a semantic model in the form of an ontology for dependability analysis with the table structure (Fang et al., 2016; Javed et al., 2013; Pahl, 2005), although a full formalisation would enhance analysis quality. A remaining challenge is the difficulty of providing domain knowledge is a suitable form. While a generic method can provide for instance metrics and can give guidance on what differentiates different measurements in terms of orders of magnitude (e.g., 0.99 vs. 0.999), these need to be linked to concretely acceptable figures that are often domain-specific. Here, we still aim to improve the method using an industrial trial.

REFERENCES

Al-Kuwaiti, M., Kyriakopoulos, N., and Hussein, S. (2009). A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability. *IEEE Communications Surveys & Tutorials*, 11(2):106–124.

Avizienis, A., Laprie, J. ., Randell, B., and Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

Barrett, R., Patcas, L. M., Pahl, C., and Murphy, J. (2006). Model driven distribution pattern design for dynamic web service compositions. In *Proceedings of the 6th International Conference on Web Engineering*, ICWE '06, page 129–136, New York, NY, USA. Association for Computing Machinery.

Barzegar, H. R., El Ioini, N., Le, V. T., and Pahl, C. (2020a). Wireless network evolution towards service continuity in 5g enabled mobile edge computing. In *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 78–85.

Barzegar, H. R., Le, V. T., El Ioini, N., and Pahl, C. (2020b). Service continuity for ccam platform in 5g-carmen. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1764–1769.

Dobaj, J., Schmittner, C., Krisper, M., and Macher, G. (2019). Towards integrated quantitative security and safety risk assessment. In *Intl Conf on Computer Safety, Reliability, and Security*, pages 102–116.

El Ioini, N. and Pahl, C. (2018). Trustworthy orchestration of container based edge computing using permissioned blockchain. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 147–154.

Fang, D., Liu, X., Romdhani, I., Jamshidi, P., and Pahl, C. (2016). An agility-oriented and fuzziness-embedded semantic model for collaborative cloud service search, retrieval and recommendation. *Future Generation Computer Systems*, 56:11–26.

Fronza, I., El Ioini, N., Pahl, C., and Corral, L. (2019). *Bringing the Benefits of Agile Techniques Inside the Classroom: A Practical Guide*, pages 133–152. Springer Singapore, Singapore.

Gand, F., Fronza, I., El Ioini, N., Barzegar, H. R., Le, V. T., and Pahl, C. (2020). A lightweight virtualisation platform for cooperative, connected and automated mobility. In *Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport*

Systems - Volume 1: VEHITS,, pages 211–220. INSTICC, SciTePress.

IEEE (2010). Ieee standard classification for software anomalies. *IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)*, pages 1–23.

ISO/IEC (2011). Iso/iec 25010:2011 – systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models.

ISO/IEC (2018). Iso/iec 27005:2018 – information technology — security techniques — information security risk management.

Javed, M., Abgaz, Y. M., and Pahl, C. (2013). Ontology change management and identification of change patterns. *Journal on Data Semantics*, 2(2):119–143.

Kenny, C. and Pahl, C. (2005). Automated tutoring for a database skills training environment. *SIGCSE Bull.*, 37(1):58–62.

Le, V. T., Pahl, C., and El Ioini, N. (2019). Blockchain based service continuity in mobile edge computing. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 136–141.

Lei, X., Pahl, C., and Donnellan, D. (2003). An evaluation technique for content interaction in web-based teaching and learning environments. In *Proceedings 3rd IEEE International Conference on Advanced Technologies*, pages 294–295.

Melia, M. and Pahl, C. (2009). Constraint-based validation of adaptive e-learning courseware. *IEEE Transactions on Learning Technologies*, 2(1):37–49.

Mellor, P. (1992). Failures, faults and changes in dependability measurement. *Information and Software Technology*, 34(10):640 – 654.

Michael, E., Woos, D., Anderson, T., Ernst, M. D., and Tatlock, Z. (2019). Teaching rigorous distributed systems with efficient model checking. In *EuroSys Conference*, pages 1–15.

Much, A. (2016). Automotive security: challenges, standards and solutions. *Softw. Qual. Prof*, 18(4):4–12.

Murray, S., Ryan, J., and Pahl, C. (2003). A tool-mediated cognitive apprenticeship approach for a computer engineering course. In *Proceedings 3rd IEEE International Conference on Advanced Technologies*, pages 2–6.

Pahl, C. (2005). Layered ontological modelling for web service-oriented model-driven architecture. In Hartman, A. and Kreische, D., editors, *Model Driven Architecture – Foundations and Applications*, pages 88–102, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pahl, C., Barrett, R., and Kenny, C. (2004). Supporting active database learning and training through interactive multimedia. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '04, page 27–31, New York, NY, USA. Association for Computing Machinery.

Pahl, C., El Ioini, N., Helmer, S., and Lee, B. (2018). An architecture pattern for trusted orchestration in iot edge clouds. In *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 63–70.

Pahl, C., Fronza, I., El Ioini, N., and Barzegar, H. R. (2019). A review of architectural principles and patterns for distributed mobile information systems. In *Intl Conf on Web Information Systems and Technologies*.

Pop, F. and Cristea, V. (2019). Distributed systems education: From traditional models to new paths of learning. In *Intl Conf on Control Systems and Computer Science*, pages 383–386.

Schoitsch, E. and Skavhaug, A. European perspectives on teaching, education and training for dependable embedded and cyber-physical systems. In *Euromicro Conference on Software Engineering and Advanced Applications*.

Scolati, R., Fronza, I., El Ioini, N., Samir, A., and Pahl, C. (2019). A containerized big data streaming architecture for edge cloud computing on clustered single-board devices. In *Proceedings of the 9th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER,*, pages 68–80. INSTICC, SciTePress.

Serpanos, D. (2019). There is no safety without security and dependability. *Computer*, 52(6):78–81.

Shan, L., Sangchoolie, B., Folkesson, P., Vinter, J., Schoitsch, E., and Loiseaux, C. A survey on the applicability of safety, security and privacy standards in developing dependable systems. In *Intl Conf on Computer Safety, Reliability, and Security*.

Verma, S., Gruber, T., Schmittner, C., and Puschner, P. (2019). Combined approach for safety and security. In *International Conference on Computer Safety, Reliability, and Security*, pages 87–101. Springer.

von Leon, D., Miori, L., Sanin, J., El Ioini, N., Helmer, S., and Pahl, C. (2019). *A Lightweight Container Middleware for Edge Cloud Architectures*, chapter 7, pages 145–170. John Wiley & Sons, Ltd.

Walter, C. J. and Suri, N. (2003). The customizable fault/error model for dependable distributed systems. *Theoretical Computer Science*, 290(2):1223–1251.