





Use-Cases for Uncertainty Propagation in Distributed Control Systems

Michael Krisper^(✉) , Johannes Iber , and Jürgen Dobaj 

Institute for Technical Informatics, Graz University of Technology,
Inffeldgasse 16/I, 8010 Graz, Austria
office@iti.tugraz.at
<http://www.iti.tugraz.at>

Abstract. This paper describes how data quality can be used to gain trust between components in distributed control systems by adding information about quality to data values. Especially numeric uncertainty is a helpful tool for making highly informed decisions. To illustrate the benefits and challenges, several use-cases are discussed in the context of industrial and automotive settings. The target audience are architects and developers of cyber-physical systems in industrial and automotive domains, researchers in such domains and software developers who are writing software for embedded or distributed control systems which also use uncertain sensor measurements.

Keywords: Quality · Uncertainty · Measurement
Error propagation · Tolerance · Predictive maintenance
Sensor-fusion · Approximate computing

1 Introduction

In the past, control systems were isolated and closed systems which have been under control of one manufacturer or closed and protected environments. This paradigm changed with the recent upcoming of ubiquitous and distributed devices, the Internet of Things (IoT), and Cyber-Physical Systems (CPS). Control systems now are often distributed and may be highly dependent on other systems which are developed by other companies. This leads to many exciting kinds of problems. In order to cope with many of those problem fields, standards were established to harmonize technical compatibility (e.g. communication, data formats, and protocols) as well as warranty, safety and contracting issues (safety and quality standards like ISO61508, ISO26262, ISO25012, AUTOSAR, ASPICE, Functional Safety, 6σ). While safety and quality standards cover the whole development and production process until delivery, the newest trend shifts the actual binding time of decisions far beyond delivery to the actual usage and runtime of a product [24, 26]. Amorim et al. describe in their papers [1, 2] how to do this by using contracts consisting of demands and

guarantees between components and evaluating them at runtime. In their contracts they depend on using attributes which can be evaluated during runtime. The challenge is now to evaluate information about the environment during runtime. In order to do this, the notion of data quality comes into play. The ISO 25012 [15] defines characteristics of data quality and in this paper, we especially look at accuracy, precision, consistency and credibility. We illustrate this in use-cases which use the measurement uncertainty of sensors as an attribute for data quality. These use-cases concern not only safety but also other dependability attributes like availability and reliability. The main idea is to evaluate the quality of a sensor via its uncertainties of measured values. This data quality attribute should be used and propagated over the whole signal path of a system in order to make informed decisions and to have more information about a systems' state available at runtime. If the sensor has small measurement errors and is not biased, the measured data has high quality and can be trusted. In such a way a consumer of a value can decide dynamically at runtime if the values are trustworthy and react based on changes of data quality e.g. when a sensor gets dirty or faulty.

We want to encourage and motivate developers and architects to be involved in this culture of quality based thinking in order to increase business value in changing environments and contexts. This corresponds to the main values and principles of the Software Process Improvement Manifesto (SPI Manifesto) [23], which is a guide for exchanging wisdom and experiences in all areas of software process improvement.

The remaining paper is structured as follows: Sect. 2 gives an overview over the related work and background. The main part of the paper focuses on the use-cases for uncertainties beginning from Sect. 3 to Sect. 9. The paper closes with a conclusion in Sect. 10.

2 Background and Related Work

Uncertainty in Measurements. According to the ISO GUM Standard “*Error is an idealized concept and errors cannot be known exactly.*” [17]. Measuring the reality is a task which always involves some kind of uncertainty. Measurement devices cannot be infinitely precise or measured objects may change, and it may even be impossible to measure everything exactly. In lack of infinitely precise measurements, we try to tackle uncertainty by equipping our sensors and values with tolerance ranges or limits. Such tolerances tell us how precise our values can be. The ISO GUM (JCGM 100:2008) standard and all additions define how to express uncertainty in measurements [17–21].

Uncertainty in Distributed Control Systems. Distributed control systems cover many areas like control systems, embedded systems, or cyber-physical systems (CPS) as they are called nowadays. They all involve some kind of uncertainty. Classic literature about this topic is e.g. “Distributed Systems” by Tannenbaum et al. [29], or “Distributed Systems Architecture” by Puder et al. [28]. According to Amorim et al. “CPS usually operate in uncertain environments and are

often safety-critical” [1]. This dangerous combination of uncertainty and safety-critical devices could lead to threatening situations causing harm, injuries and the death of humans as well as substantial damage of properties and financial losses. According to the HAZOP model, data errors in control systems are categorized as *Provision Errors*, *Timing Errors*, and *Value Errors* [4, 11, 12]. To handle these kinds of errors, contracts with requirement definitions can be used [3]. For example, in the ConSerts M model [1, 2], every component provides guarantees to the consuming component, while requesting demands to the serving components. In such a way it is possible to define safety contacts at design time which are evaluated at runtime and acted upon via safety mechanism through self-adaptive systems [14].

Uncertainty in Probabilistic Programming. Basically every measured value can be represented as random variable, which is done in probabilistic programming. Such random variables have continuous or discrete probability distributions which are used for inference, arithmetic and conditionals in a program. They have some generic mechanisms in common [25]: (i) *a probabilistic model*, (ii) *propagation rules*, and (iii) *inference techniques*. Andy Gordon wrote a survey about the current state of probabilistic programming [13]. Bornholt et al. published several papers about their implementation approach of such mechanisms [5–7]. Another aspect of this is approximate computing, which creates software with just enough precision as needed. Darulova et al. investigated many aspects of approximate computing and created a framework for compiling programs with uncertainties, to be faster and use less memory [8–10, 16].

3 Use-Case 1: Quality Evaluation Based on Uncertainty

“Researchers in computer systems either do not know about measurement bias or do not realize how severe it can be.” [27].

Using uncertainty for measured values allows for informed decisions, better evaluation of the environment and sophisticated safety arguments, which can contain lower and upper bounds for safety margins (Fig. 1).

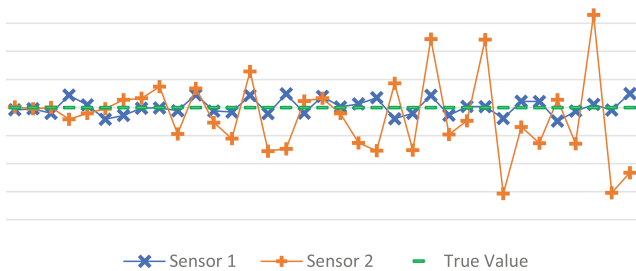


Fig. 1. Quality evaluation based on uncertainty in Sensor Data: which of the two sensors is more trustworthy?

Very often decisions and calculations are done with data directly coming from a sensor or other systems, and are trusted to be 100% correct. Contracts and design arguments protect us from getting biased or uncertain data, but do we really know for sure even during runtime? Mytkowicz et al. [27] showed that experimental measurements of computer systems regarding performance are always flawed by not using diverse environments and could potentially lead to wrong claims by not considering measurement bias. That is why they proclaim a call to action to consider measurement bias in computer systems. While this mainly applies to experimental measurements, it is also important for sensors, which exhibit even more uncertainty, which can't be reproduced as easily as in experimental environments. In safety, redundancy and diversification are key concepts for reducing failure rates and common cause failures. We propose to expand these concepts to allow better informed decisions about systems.

Wrong decisions potentially could lead to endangering human life, harm and injuries, but also enormous financial damage. For example, autonomous self-driving cars constantly monitor the environment and decide in adaptive control-loops which action is the most appropriate. Such decisions are guided by data from multiple sensors in order to drive safely and avoid accidents.

This use-case exhibits following forces or challenges:

- Sensor Data is always uncertain (as is every measurement). Therefore, it could be inaccurate, and without modeling these uncertainties this could lead to wrong assumptions and decisions.
- Exact tolerances are often unknown. Of course, you could assume the worst-case tolerances from the data sheet of a sensor, but oftentimes they are way overrated, and still one cannot guarantee during runtime that they are still satisfied.
- Decisions based on inaccurate or oversimplified data could lead to wrong results (injuries, fatalities, ...). Assuming a measured value is infinite precise is very dangerous and careless.

The goal in the context of this use-case is to make safe and informed decisions with the help of error-margins and safety assumptions to avoid and mitigate injuries and erroneous behavior. For this, we need systems which have mechanisms for defining uncertainties, for propagating them, and finally for decision-making with known guarantees and confidence.

4 Use-Case 2: Predictive Maintenance

The evaluation of uncertainty and measurement tolerances could potentially increase the prediction accuracy for predictive maintenance. By establishing degradation models which reason about how failure and quality of a component are related, a manufacturer can predict how long the product lifetime will be based on the current state of quality in the product evaluated during runtime. In such a way it would be possible to avoid unnecessary maintenance efforts, but

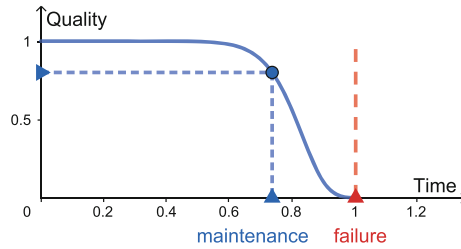


Fig. 2. Usage of quality information for predictive maintenance: as soon as the quality of the sensor is under a predefined limit, maintenance actions should be done before the component fails.

just replacing or reconditioning those parts where and when the attention really is needed (Fig. 2).

Think of the trivial example of motor oil. We have long periodic maintenance intervals because modern engines are working much cleaner as ever before. Nevertheless, motor oil is still replaced in regular intervals (either time span or driven distance) although it may not be necessary in many cases. A sensor which measures the viscosity or contamination of the oil could give feedback about its state and inform the driver when it is time to be changed.

The challenges in this context are:

- Sensors may get damaged, polluted and fail over time. The precision and quality of the produced data also decreases with such decay processes.
- Periodically scheduled maintenance may be inefficient, because parts could still be fully functional even after some time and would be replaced prematurely.
- The other side of periodic maintenance is when parts fail or decay earlier than the cycle has foreseen this may go unnoticed until the maintenance time. In such cases maintenance should have been earlier to ensure correct functionality of the components.

The goal in this use-case is to save costs for unnecessary maintenance while ensuring that all safety goals are met and the functionality of all components is ensured. Just doing maintenance when it is really needed has many advantages to the whole ecosystem of products. The predictive maintenance model and its evaluation during runtime also has another very beneficial side effect: By continuously monitoring the health state of the components we can detect early or unexpected failures during runtime. By knowing and monitoring the quality level (especially the sensor tolerances) one can predict the failure rate more accurately during runtime. By using a model for failure-rates based on the runtime tolerances we can predict the point in time when the sensor will fail. Based on that, maintenance should be planned.

5 Use-Case 3: Sensor Fusion

Use data quality, e.g. uncertainty, to combine several input values in order to get results with even higher quality, accuracy and less uncertainty. Also, use it to give the most accurate data more weight than the inaccurate (Fig. 3).

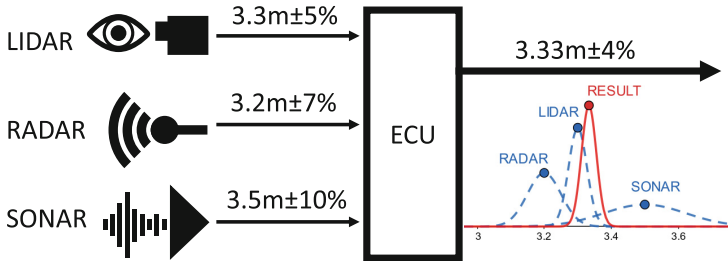


Fig. 3. Sensor fusion by combining the individual sensor value and their according probability distributions. The result has higher quality and lower uncertainty than every single sensor.

Sensor fusion is a huge area in control systems which is researched for many decades now but still makes huge advancements when it comes to new technologies and how to combine them. For more information about Sensor Fusion we propose the book by Klein: “Sensor and data fusion: a tool for information assessment and decision-making” [22]. In this use-case we concentrate on the fusion of data signals which are semantically similar (measuring the same information), e.g. distance, energy consumption, or signal strength. The only things which differ for similar measurements are the actual measured value, and the uncertainties exhibited by the sensor and the measurement. This makes it easier to combine the values by just combining their respective measurements seen as random variables with uncertainties. Equations for combining multiple independent input values with normal distributions (μ is the expected value or mean, σ is the standard deviation):

$$X_0, \dots, X_n \sim \mathcal{N}(\mu_i, \sigma_i^2) : \hat{X} = \frac{1}{n} \sum X_i \sim \mathcal{N}\left(\frac{1}{n} \sum \mu_i, \frac{1}{n^2} \sum \sigma_i^2\right)$$

It is noteworthy, that for the above equations we assumed independent random variables. This is not the case all the time, because the sensors try to measure the same “true” value, and are therefore correlated to each other. If the “true” value changed, all sensor values are expected to change accordingly – therefore, they are dependent and for sensor fusion also the covariances of the sensors should be considered. Another assumption here is that all sensor uncertainty is described using normal distributions for their measurements. When they have different distributions, this should be considered accordingly in order to maintain their probability properties.

The challenges in this context are:

- A single sensor may be inaccurate and its quality may change during usage over time.
- Computing simple averages amongst multiple sensor hides the uncertainty coming from the sensors and therefore gives a wrong image of certainty.
- Better and worse data are mixed together, which should be accounted accordingly.

The goal is to use data from multiple sensors to get more accurate information about the environment. The combined value should exhibit lower uncertainty than any single sensor value. Therefore, it is needed to evaluate the quality of the sensors during runtime and use this quality information for the sensor-fusion.

We propose to weight the sensors according to their uncertainty, in order to prefer more accurate sensors over the ones which are imprecise. This has two highly beneficial consequences:

1. **Environmental Adaption:** When one sensor is better for near distance measurements (e.g. in low-speed situations) and another is better at far distance measurements (e.g. high-speed situations), weighting them according to their precision would result in an automatic adaption to the current environment and always using the best source of data for a given situation.
2. **Failover:** In cases where a sensor fails completely, it can be completely overruled by the still functional working sensors, because its uncertainties would get very high and therefore its value would be weighted very low. This would result in failover situation where the system still continues to function, despite a sensor failing. Of course, this only is possible for systems which are designed to have redundant signal paths or even diverse sensors, in order to avoid common cause errors. Amorim et al. described an architecture which makes use of alternative data sources in case of failures [2].

6 Use-Case 4: Approximate Computing

Perform calculations only with the needed precision to increase performance and save memory.

Figure 4 shows the time needed for calculating pi with a variant of the Gregory-Leibniz Series ($\pi = 4 \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}$). The figure shows that for increasing the accuracy (decreasing the error) much more time is needed. For example, if we need the value to have a precision of 10^{-3} , the algorithm only needs about 0.5 ms, but if we need a precision of 10^{-4} , we would have to let it run for 5 ms (10 times longer). Of course, in this case, there are much faster methods available, but it shows how beneficial approximate computing could be for algorithms which do not have a fast alternative. By aborting the calculation as soon as the needed precision is reached one can save much computing time [8].

In addition to performance also memory could be saved by using approximations. Many applications use double or float as data types for their floating-point variables, but only need precision of a few decimal places. These could be replaced by fixed point arithmetic which perform much faster while still supplying the needed precision [8–10, 16]. The challenges in this context are:

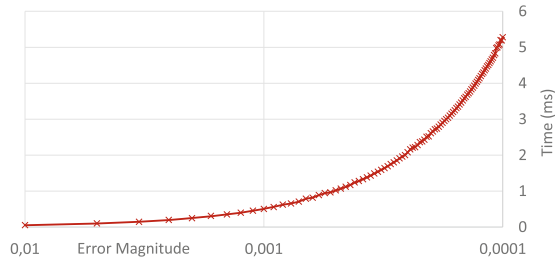


Fig. 4. Performance benchmarks for calculating π with the Gregory-Leibniz Series. To increase the accuracy of the solution, much more computing time is needed.

- Exact calculations often take unnecessary long time.
- Approximate calculations may be faster, but you need to balance the precision to your needs.

For this use-case the goal is to speed up calculations and save memory by using approximations which only use as much preciseness as is needed. In order to do that, uncertainty information comes in handy because it could be applied in two ways: Firstly, the maximum precision depends upon the precision of the input values, respectively the input sensors. It does not make sense to apply more exact algorithms when the uncertainty of the input data is already very high. Secondly, the needed precision in calculations depends upon the ultimately required precision of the output value. For example, it would be futile to numerically optimize some algorithms to the 10^{-6} decimal place, while the calculated output value is then rounded to whole integer numbers.

7 Use-Case 5: Fault Detection

Use quality information (e.g. uncertainty or standard deviation), to detect additional faults in components which would go unnoticed otherwise.

Using quality information like accuracy or uncertainty gives the possibility to define additional checks for fault detection. Thresholds on the *quality* of a signal can be defined in addition to the range-checks which are defined at design time according to the data-sheet or interface description of a component (e.g. the HSI: Hardware-Software-Interface-Specification) (Fig. 5).

The challenges are:

- Sensor Quality (e.g. tolerances) may change over time.
- Safety functions rely on good quality information to e.g. apply boundary or threshold checking.
- Tolerances coming from data sheets may be exaggerated and represent maximum values, which leads to very conservative assumptions.

The goal of this use-case is to have tolerance information available during runtime in order to be used for safety functionality and to detect faulty and

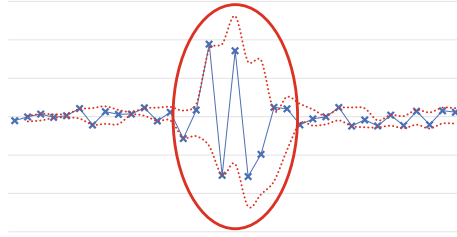


Fig. 5. Detect faults by evaluating the uncertainty of a value series.

maybe even harmful situations. When a system would know the tolerances and uncertainties of the used signals and sensors at runtime, it could easily detect when something goes out of bounds, or when tolerances of a value suddenly increase without any reason. This demands that the sensor quality is measured periodically in order to have recent information available to guarantee the liveness and correctness of the error margins.

8 Use-Case 6: Fingerprinting

Use quality information as fingerprints to identify individual components (Fig. 6).

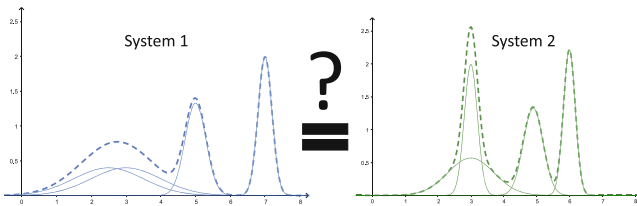


Fig. 6. Identify systems based on their individual calibration or uncertainty profile.

During production the uncertainties of sensors are measured and their calibration is set. This configuration of calibration data is a set unique to each system. By utilizing the initial calibration and uncertainty information one can calculate a unique fingerprint for the identification of a system and use this as an identification later on. The challenges of such a use case are:

- Devices and Sensors cannot be trusted a priori.
- Devices and Sensors in combination have a pretty unique configuration of calibration data, uncertainties and tolerances.
- Authentication mechanism need additional functionality (TPM, certificates, identities, key-exchange) which may be too expensive.

The goal is to identify sensors and devices according to their fingerprints without having to implement additional security features or hardware. This can be done by utilizing the uncertainty and calibration data of the systems' sensors during production and storing this profile information as fingerprint. During runtime measure the system again and compare to the stored fingerprint. If the profile is mostly the same, this is a strong indicator that the measured system actually is the same.

9 Use-Case 7: Graceful Degradation

Degrade functionality of a system based on the quality of the sensors and sophisticated safety assumptions (Fig. 7).

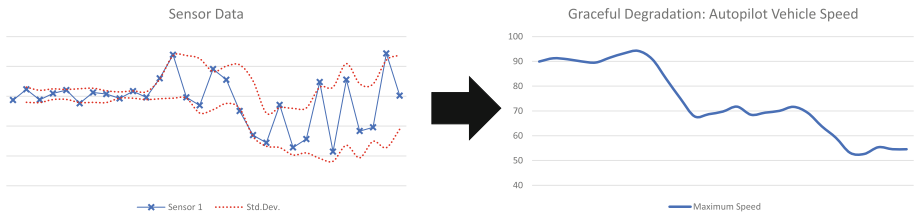


Fig. 7. Depending on the quality of the sensor input data, an autopilot can regulate the maximum speed for a degraded state.

Graceful degradation is a technique which is applied as safety measure if a system's functionality must be ensured but cannot be fully supplied. There can be two reasons why a degraded system is valuable: Firstly, when the system has to go into a safe-state, this oftentimes cannot be done immediately, but has to degrade gracefully over time, so that the driver or user can accustom to the new situation. Secondly, it is often preferred to have at least a degraded functionality than no functionality at all. Amorim et al. wrote about graceful degradation and how it can be applied to situations when the contracts are not fulfilled at runtime [2]. They depicted the situation where a sensor fails to operate and their solution was to search for other data sources which can be used despite the possibility that they may be more inaccurate. If the other inputs can't provide the needed ASIL level, the car should still be controllable, but in degraded mode in order to minimize possible hazards (the maximum speed is reduced).

Despite graceful degradation, also graceful improvement would be possible: if the uncertainty and data quality gets better, a system could adaptively increase the functionality. If the constraints are still met due to better sensors with smaller measurement tolerances, the maximum speed could even be increased while maintaining the same safety level.

10 Conclusion

In this paper we showed seven use-cases where the use of uncertainty as indicator for data quality is very beneficial for the dependability of a system. Trust in the sensors, the data, and the whole signal path can be increased by evaluating data quality of the numerical values. Especially using numerical uncertainty is helpful in making highly informed decisions which could potentially save lives. In the future, we plan to investigate each use-case in detail and find appropriate techniques and integration possibilities for existing systems in real life projects and scenarios. In the spirit of the SPI manifesto [23] we want to motivate and encourage manufacturers, developers and software as well as system architects to apply uncertainty and quality considerations in their systems to change their daily business for the better.

References

1. Amorim, T., et al.: Runtime safety assurance for adaptive cyber- physical systems: ConSerts M and ontology-based runtime reconfiguration applied to an automotive case study. In: Solutions for Cyber-Physical Systems Ubiquity, pp. 137–168. IGI Global (2018). <https://doi.org/10.4018/978-1-5225-2845-6>
2. Amorim, T., Ruiz, A., Dropmann, C., Schneider, D.: Multidirectional modular conditional safety certificates. In: Koornneef, F., van Gulijk, C. (eds.) SAFECOMP 2015. LNCS, vol. 9338, pp. 357–368. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24249-1_31
3. Beugnard, A., Jézéquel, J.M., Plouzeau, N.: Contract aware components, 10 years after. Electron. Proc. Theor. Comput. Sci. **37**, 1–11 (2010). <https://doi.org/10.4204/EPTCS.37.1>
4. Bondavalli, A., Simoncini, L.: Failure classification with respect to detection. In: Proceedings of the Second IEEE Workshop on Future Trends of Distributed Computing Systems, pp. 47–53, September 1990. <https://doi.org/10.1109/FTDCS.1990.138293>
5. Bornholt, J., Meng, N., Mytkowicz, T., McKinley, K.S.: Programming the internet of uncertain <T>hings, pp. 1–7 (2015)
6. Bornholt, J., Mytkowicz, T., McKinley, K.S.: Uncertain<T>: a first-order type for uncertain data, p. 21 (2013)
7. Bornholt, J., Mytkowicz, T., McKinley, K.S.: Uncertain<T>: a first-order type for uncertain data, pp. 51–66. ACM Press (2014). <https://doi.org/10.1145/2541940.2541958>
8. Darulova, E.: Programming with numerical uncertainties, p. 172 (2014)
9. Darulova, E., Kuncak, V.: Sound compilation of reals, pp. 235–248. ACM Press (2014). <https://doi.org/10.1145/2535838.2535874>
10. Darulova, E., Kuncak, V., Majumdar, R., Saha, I.: Synthesis of fixed-point programs, pp. 1–10. IEEE, September 2013. <https://doi.org/10.1109/EMSOFT.2013.6658600>
11. Ezhilchelvan, P.D., Shrivastava, S.K., Elphick, M.J.: A characterisation of faults in systems. University of Newcastle upon Tyne, Computing Laboratory (1985)
12. Fenelon, P., Hebronn, B.: Applying HAZOP to software engineering models. In: Risk Management and Critical Protective Systems: Proceedings of SARSS 1994 (1994)

13. Gordon, A.D., Henzinger, T.A., Nori, A.V., Rajamani, S.K.: Probabilistic programming. In: Proceedings of the on Future of Software Engineering, FOSE 2014, pp. 167–181. ACM, New York (2014). <https://doi.org/10.1145/2593882.2593900>
14. Iber, J., Rauter, T., Kreiner, C.: A self-adaptive software system for increasing the reliability and security of cyber-physical systems. In: Solutions for Cyber-Physical Systems Ubiquity, pp. 223–246 (2018). <https://doi.org/10.4018/978-1-5225-2845-6.ch009>
15. ISO, IEC: ISO/IEC 25012:2008 data quality model (2008)
16. Izycheva, A., Darulova, E.: On sound relative error bounds for floating-point arithmetic, pp. 15–22. IEEE, October 2017. <https://doi.org/10.23919/FMCD.2017.8102236>
17. JCGM: JCGM 100:2008 evaluation of measurement data Guide to the expression of uncertainty in measurement, September 2008
18. JGCM: JCGM-WG1-SC1-N10 guide to the expression of uncertainty in measurement (GUM) - supplement 1: numerical methods for the propagation of distributions (2004)
19. JGCM: JCGM 104:2009 evaluation of measurement data - an introduction to the “guide to the expression of uncertainty in measurement” (2009)
20. JGCM: JCGM 102:2011 evaluation of measurement data - supplement 2 to the “guide to the expression of the uncertainty in measurement” - extension to any number of output quantities (2011)
21. JGCM: JCGM 106:2012 evaluation of measurement data - the role of measurement uncertainty in conformity assessment. Chem. Int. - Newsmagazine IUPAC **35**(2) (2013). <https://doi.org/10.1515/ci.2013.35.2.22>
22. Klein, L.A.: Sensor and data fusion: a tool for information assessment and decision making. SPIE Press, Bellingham (2004)
23. Korsaa, M., et al.: The SPI manifesto and the ECQA SPI manager certification scheme. J. Softw.: Evol. Process **24**(5), 525–540 (2012). <https://doi.org/10.1002/smr.502>
24. Kreiner, C.: A binding time guide to creational patterns, pp. 1–10. ACM Press (2015). <https://doi.org/10.1145/2739011.2739025>
25. Krisper, M., Iber, J., Dobaj, J., Kreiner, C.: Uncertain values, error-propagation, and decision confidence, p. 5. ACM, Irsee (2018, unpublished)
26. Krisper, M., Kreiner, C.: Describing binding time in software design patterns, pp. 1–15. ACM Press (2016). <https://doi.org/10.1145/3011784.3011811>
27. Mytkowicz, T., Diwan, A., Hauswirth, M., Sweeney, P.F.: Producing wrong data without doing anything obviously wrong! p. 12 (2009)
28. Puder, A., Römer, K., Pilhofer, F.: Distributed Systems Architecture: A Middleware Approach. Elsevier, Amsterdam/Boston (2006)
29. Tanenbaum, A.S., van Steen, M.: Distributed Systems: Principles and Paradigms, 2nd edn. Pearson Education, Harlow/Essex (2014)