The Next Frontier

Reliability of Complex Systems

Schenkelaars, D.; van Driel, Willem Dirk ; Duijve, R.

DOI
10.1007/978-3-319-58175-0_22

Publication date
2018

Document Version
Final published version

Published in
Solid State Lighting Reliability Part 2

Chapter 22
The Next Frontier: Reliability of Complex Systems

D. Schenkelaars, Willem Dirk van Driel, and R. Duijve

Abstract Traditional lighting is focused on the prevention of hardware failures. With the trend toward controlled and connected systems, other components will start playing an equal role in the reliability of it. Here reliability need to be replaced by availability, and other modeling approaches are to be taken into account. Software reliability can only be covered by growth models, with the Goel-Okumoto as a promising candidate. System prognostics and health management is the next step to service the connected complex systems in the most effective way possible. In this chapter we highlight the next frontiers that will need to be taken in order to move the traditional lighting catastrophic failure thinking into a thinking more toward new ways how system (degraded) functions can fail or be compromised.

22.1 Introduction

Nowadays the lighting industry experiences an exponential increasing impact of digitization and connectivity of its lighting systems [1]. The impact is far beyond the impact on single products but extends to an ever larger amount of connected systems. Continuously, more intelligent interfacing with the technical environment and with different kind of users is being built-in by using more and different kinds of sensors, (wireless) communication, and interacting or interfacing devices. Figure 22.1 gives two examples toward these controlled and connected systems, just to highlight the scale of it.

When the number of components and their interactions significantly increase, so-called large or complex systems are formed. The commonly used description of a large or complex system is given as [2, 3]:

D. Schenkelaars (✉) • R. Duijve
Philips Lighting, HTC45, 5656 AE, Eindhoven, The Netherlands
e-mail: dick.schenkelaars@philips.com; r.duijve@philips.com

W.D. van Driel
Philips Lighting, High Tech Campus, Eindhoven, The Netherlands

Delft University of Technology, EEMCS Faculty, Delft, The Netherlands
e-mail: willem.van.driel@philips.com

W.D. van Driel et al. (eds.), *Solid State Lighting Reliability Part 2*, Solid State Lighting Technology and Application Series 3, DOI 10.1007/978-3-319-58175-0_22

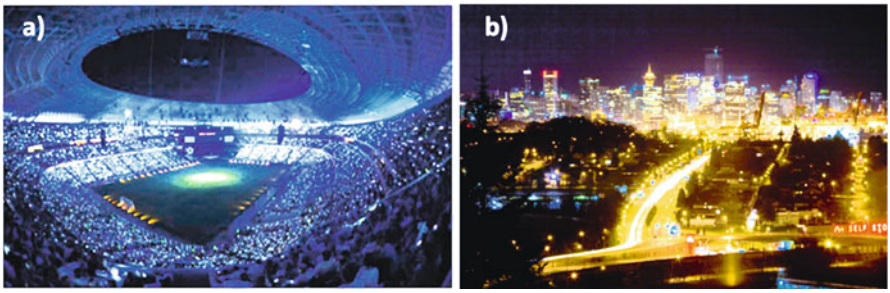

Fig. 22.1 Two examples of controlled and connected systems, with (**a**) >1000 connected luminaires on one theater system and (**b**) >10,000 connected street luminaires in one city

A complex system: *a system composed of interconnected parts that as a whole exhibit one or more properties (behavior among the possible properties) not obvious from the properties of the individual parts.*

With the increasing amount of complexity, it is imperative that the reliability of such systems will enter a next frontier. In this chapter we will discuss the current state of the art and challenges that are to be confronted in order to tackle this.

22.2 All Components Matter

The functions in a complex lighting product can be listed as four basic properties [1, 3] being the (i) traditional lighting unit and its components, (ii) the software needed for processing data, (iii) a monitoring function for getting this data, and (iv) the communication to, for example, the user or the product. Examples of these functions are listed below:

(i) Lighting unit components

- Hardware: electronics, LED, PCBs, optics, plastics, solders
- Connectors – indoor & outdoor – , rigid and flexible
- Other mechanical connectors (e.g. screws, clips)
- Moving parts
- Fans, SynJet, and motor drives (air cooling)
- Wires
- Batteries
- Sealants

(ii) Processing and storage

- Software
- Data storage

Table 22.1 Complexity level versus reliability fingerprint

System complexity	High-level reliability fingerprint
Lighting unit components	Catastrophic failures of sub-components, relative well known
Controlled systems	More functions, more complex failure modes, software failure as component
Connected lighting systems	Degradation of system features and/or nuisance incidents with poor predictability, strong impact of software, availability, and customer experience

(iii) Monitoring

- Sensors – indoor and outdoor
- IoT – Internet of Things

(iv) Communication

- Sensors – indoor and outdoor
- Wireless/wired connectivity (including data integrity)
- IoT – Internet of Things

When processing and monitoring is added, one creates a so-called controlled lighting system. By adding the communication part, the system further evolves into a connected lighting system. Obviously, the system complexity increases in these three steps, simply because more components are added (see the above definition). Each of these systems has its own fingerprint when it concerns reliability, see Table 22.1 for a high-level view. On the lowest level of complexity, catastrophic failure of the sub-components determine the reliability and lifetime of the system. When the system is controlled, more complex failure modes may occur, and software failure is added to the equation. In the highest possible level of complexity, a fully connected and controlled lighting system, it becomes more difficult to predict the reliability performance, and here it is availability that come into play. In order to capture the systems availability, it is essential to introduce data analytics. In the following paragraphs, we will discuss these topics in more details.

22.3 Complex Systems: Availability Rather Than Reliability

System availability is the degree to which a system is operational and accessible, when required for use [4]. It can be defined as the part of a system that is functional (over time) following the below equation:

$$A = 1 - \frac{\#\ \text{of failing components}}{\text{total}\ \#\ \text{components}} \tag{22.1}$$

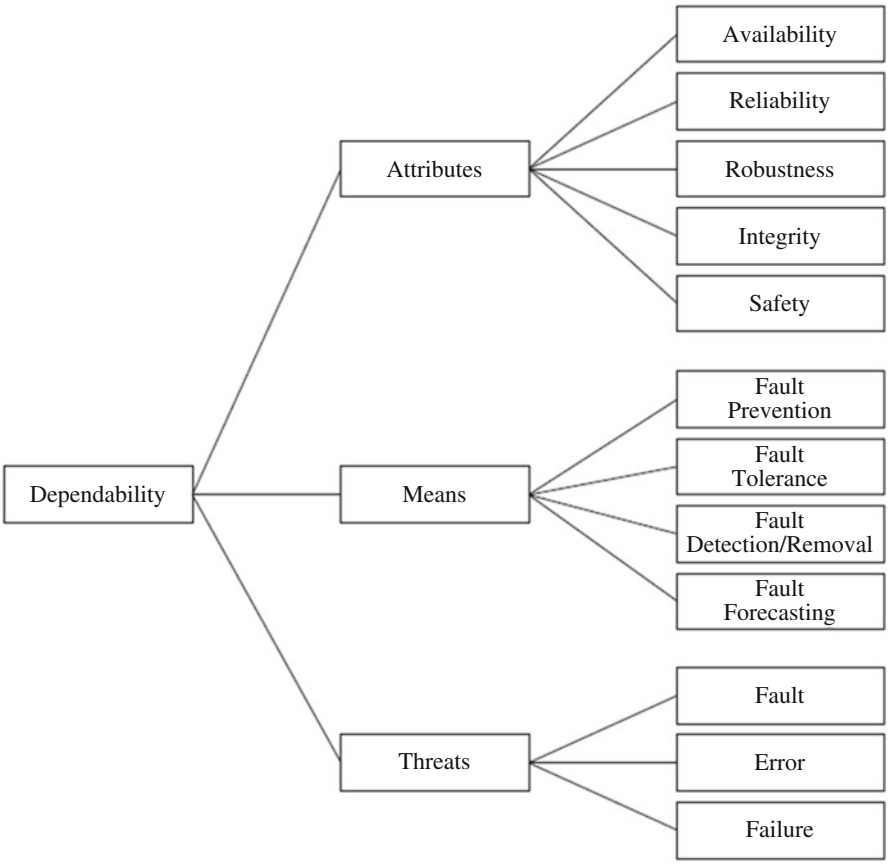

Fig. 22.2 The dependability tree: a measure of system's availability, reliability, and maintainability

When parts of a system are independent, availability does not scale with system size. In general, a high reliability will always lead to a high availability, but a system with low reliability can still have a very high availability. Reliability and availability belong to the attributes of the system dependability as well as integrity and safety as shown in Fig. 22.2 (input used from [5]). Dependability then is the ability of a system to avoid failures that are more frequent and more severe than acceptable. A dependable system is having all its required properties and does not show failures.

Typically for a networked and connected lighting system, this availability differs from the classical definition of system availability where system availability is defined as the fraction of time that a system provides the service for which it is specified. Where for one light point, reliability states the probability for survival after a specific period over time, for thousands of light points connected together this claim as isolated statement is not useful anymore. When thousands of light points are connected, it makes no sense to define system, or network failure, as failure of just one single light point in the system. It makes more sense to define

light point availability, indicating the fraction of light points operating in the controlled network over time. Where the formal system availability depends on planning and duration of repair, the availability of light points can be referred as "without repair" or "including repair." Repair here should be seen as system maintenance. In connected lighting systems, the availability of light points not only depends on the lighting units reliability but also on the reliability of sensors, controllers, communication devices, gateways, routers, and any other component that is in the system. Also it will depend on the robustness of software and on the system architecture which can determine how "deep" a hardware unit failure or software fault impacts the lighting system.

22.4 Testing and Validation

The creation of testing tools starts with a proper understanding of ways to quantify and predict the reliability of these systems in various complex operating environments. There are different ways of testing the system, e.g.:

- *System Testing*: The process of testing an integrated hardware and software system to verify that the system meets its specified requirements. It is conducted by the testing teams in both development and target environment.
- *Scalability Testing*: Part of the battery of nonfunctional tests which tests a software application for measuring its capability to scale up – be it the user load supported, the number of transactions, the data volume, etc. It is conducted by the performance engineer.
- *Performance Testing*: Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.
- *Compatibility Testing*: Testing technique that validates how well software performs in a particular hardware/software/operating system/network environment. It is performed by the testing teams.
- *Operational Testing*: Testing technique conducted to evaluate a system or component in its operational environment. Usually it is performed by testing teams.
- *Model-Based Testing*: The application of model-based design for designing and executing the necessary artifacts to perform software testing. It is usually performed by testing teams.
- *Acceptance Testing*: Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is usually performed by the customer.
- *Inter-systems Testing*: Testing technique that focuses on testing the application to ensure that interconnection between application functions correctly. It is usually done by the testing teams.

Testing is an essential part of any complex system that combines hardware with software and connectivity. The test effort becomes harder with an increase of the system complexity and with the number of variations that can be out in the field.

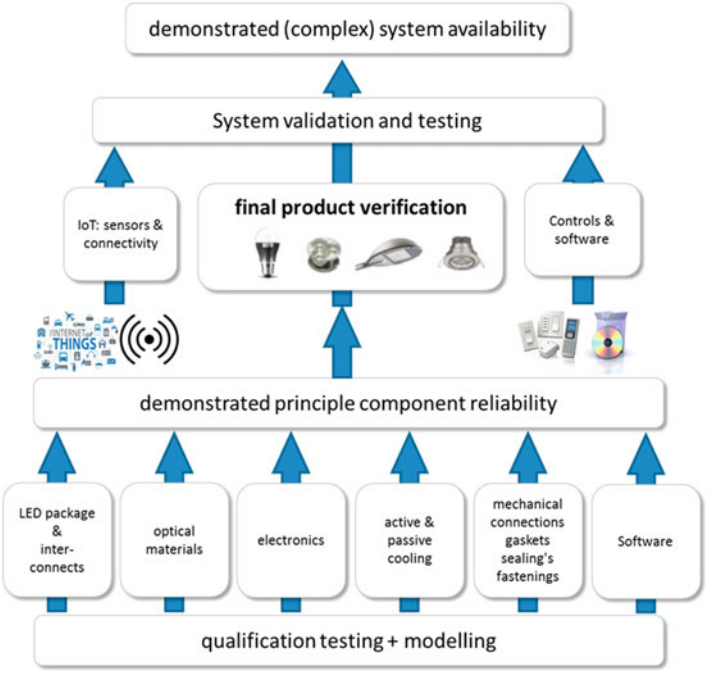

Fig. 22.3 Adding to the complexity in lighting systems: controls, software, sensors, and connectivity

Particularly if distributed connected systems reconfigure themselves in runtime, this poses challenges. For example, in identifying the proper key performance indicators (KPI) and collecting the respective critical to quality (CTQ) variables that carry the information which will allow the system to perform according to specification.

Following the approach from IEC62861, the work to create a so-called guide to principal component reliability testing for LED light sources and LED luminaires (see also Chap. 1 and [6]), new components have to be added that cover the complexity of the connected lighting system. Figure 22.3 schematically shows the added components, controls, software, and connectivity (including sensors). Testing and validation on the highest possible system level is based on the demonstrated behavior of its components. Note that also software should be seen as a component and release as such.

22.5 Software Reliability

There are many differences between the reliability and testing concepts and techniques of hardware and software. Software reliability or robustness is the probability of failure-free software operation for a specified period of time and environment

[7]. In this sense, software failures are considered a primary cause of product reliability problems, and hence a reasonable KPI for testing the software reliability is the number of software failures left in the system [8].

A software failure mode and effect analysis (FMEA) can determine the software failure modes that are likely to cause failure events [9–11]. It determines what single or multiple point failures could produce these top-level events. Software FMEAs are useful when designing or testing the error handling part of your software. Software FMEAs are also needed in order to develop inspection criteria for requirements, design, and code that are geared toward the appropriate failure modes. Design reviews are more effective when you know in advance the types of failure modes that are most likely.

Unlike hardware failures, software systems do not degrade over time, unless modified and software failures are not caused by faulty components, wear-out, or physical environment stresses such as temperature and vibration [7]. Software failures are caused by latent software defects that were introduced into the software as it was being developed but were not detected and removed before the software was released to customers. The best approach to achieving higher software reliability is to reduce the likelihood that latent defects are in released software. Unfortunately, even with the most highly skilled software engineers following industry best practices, the introduction of software defects is inevitable due to the inherent complexities of the software functionality and its execution environment.

A comparison of software and hardware reliability is useful in developing software reliability models. Table 22.2 (input used from [8]) shows the differences and similarities between the two.

Software reliability growth models (SRGM) are mathematical functions that describe fault detection and removal phenomenon [7, 8]. Some realistic issues such as imperfect debugging and learning phenomenon of software developers are incorporated in software reliability assessment. Among all SRGMs, a large class of stochastic reliability models is based on a nonhomogeneous Poisson process. These models are known as NHPP reliability models and have been widely used to track reliability improvement during software testing. Another popular class is the class of general order statistics, or GOS models. Goel-Okumoto is the most well-known NHPP model. Due to the important role that this model has played on the software reliability modeling history, it is often called "the" NHPP model. The mean value function is given by below formula:

$$m(t) = a\left(1 - e^{-bt}\right) \tag{22.2}$$

for all $t \geq 0$, where $a > 0$ and $b > 0$. The parameter a is the expected number of failures to be eventually detected while b is the rate at which each individual failure will be detected during testing. Following the sequence of software testing and

Table 22.2 Comparison between hardware and software reliability

Hardware	Software
Hardware failures are induced by component wear-out or stress invoked by thermal cycling, surges, ESD etc.	Software failures are not introduced by wear-out or stress. Software failures may be due to errors, ambiguities, oversights, or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software, or other unforeseen problems
Early failures	Software reliability is not a function of time
Constant failure rate (FIT)	No wear-out
End-of-life failures	Software will not change in time
Environmental conditions can be specified (indoor/outdoor, global area)	Errors will be induced by environments or contexts unforeseen in the design
Failure rate has a bathtub curve	Without considering program evolution, failure rate is statistically nonincreasing
Material deterioration can cause failures even though the system is not used	Failures never occur if the software is not used
Failures are caused by material deterioration, random failures, design errors, misuse, and environment	Failures are caused by incorrect logic, incorrect statements, or incorrect input data. This is similar to design errors of a complex hardware system
Hardware reliability can be improved by better design, better material, applying redundancy, and accelerated life testing	Software reliability can be improved by increasing the testing effort and by correcting detected faults
Hardware repairs restore the original condition	Software repairs establish a new piece of software
Hardware failures are usually preceded by warnings	Software failures are rarely preceded by warnings
Hardware might fail due to unforeseen application conditions	Software might (also) fail due to unforeseen application conditions

faults found, one can derive the maturity growth of the software by following this approach; Fig. 22.4 see for an example.

22.6 Reliability and Data Analytics

Traditional lighting is shifting toward connected lighting, and as a result companies are also enabled to shift more toward an information-based environment [12]. The use of information from connected sources can be described as a revolution named big data. With big data, data analytics from live connections of "intelligent" systems can be used to determine the system prognostics. Due to these changes in technology, the next generation of product data will be much richer in information [13, 14]. Reliability and availability will become enablers for product designs. Big

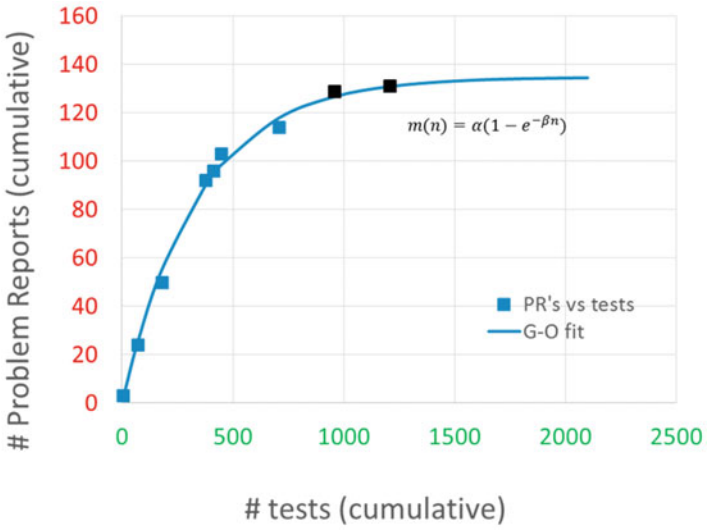

Fig. 22.4 Example of a maturity growth analysis for assessing the software reliability

data will bring detailed understanding of failure mechanisms, usage scenarios, technology, and optimal designs. For example, products can be outfitted with sensors that can be used to capture information about how and when and under what environmental and operating conditions products are being used. But the data can also be used for pure reliability analysis. Examples are signal-detection algorithms to detect unsafe operating conditions or precursors to system failure that can be used to protect a system by shutting it down or by reducing load to safe levels. And on top of this, there can be a need to predict the remaining life of the system (or the remaining life of its most important life-limiting components). This topic is named as prognostics and health monitoring (PHM). PHM refers to the process of predicting the future reliability or determining the remaining useful lifetime of a product by assessing the extent of deviation or degradation of a product from its expected normal operating conditions [15]. Today, we predict failure rates on system level following classical reliability approaches, where standardized testing and experimental failure analysis are used in order to obtain conservative bounds from the failure models. However, except in the case of reliability "incidents," there is only limited feedback with which we can judge the effectiveness of our reliability approach. Prognostics and monitoring is not just about creating a more reliable product: it is about creating a more predictable product based on real-world usage conditions. Data analytics is a necessary part of this but, in itself, is not sufficient. In order to add value, product insights need to be leveraged into the technologies that are used in order to differentiate from others. Prognostics and monitoring is not about troubleshooting reliability issues; rather, it is a new control point enabled by the transition to a lighting services business. It is the combination of data and deep physical (and technological) insight that will give a unique "right to win" in the

lighting industry. The future possibilities for using big connected data in reliability applications are unbounded. Lifetime models that are based on this data have the potential to explain much more variability in field data than has been possible before.

22.7 Final Remarks

The discussion and thinking so far in the lighting industry is focused on hardware failures. But controlled and/or connected lighting systems contain much more components; they encompass a level of complexity that goes beyond the traditional hardware thinking. Processing, monitoring, and communication functions are added with software playing an increasingly important role. Correspondingly, system reliability and availability will be increasingly dependent on the reliability of software. On top of this, the requirement of prognostics and health management is inevitable to be able to maintain and service the connected complex systems in the most effective way possible.

References

1. D. Schenkelaars, W.D. van Driel, M. Klompenhouwer, I. Flinsenberg, R. Duijve, Towards Prognostics & Health Management in Lighting Applications, European Conference Of The Prognostics And Health Management Society 2016, open access journal, available at: http://www.phmsociety.org/node/2090/, Volume 7, Page count: 7, (2016)
2. C. Joslyn L. Rocha. Towards semiotic agent-based models of socio-technical organizations, Proc. AI, Simulation and Planning in High Autonomy Systems (AIS 2000) Conference, Tucson, Arizona, (2000), pp. 70–79
3. X.J. Fan, W.D. van Driel, *Solid State Lighting Reliability: Components to Systems* (Springer, New York, 2012)
4. P.Jaramillo, A. Pruteanu, W.D. van Driel, J-P. Linnartz, Tools and methods for validation and verification, in *Runtime Reconfiguration in Networked Embedded Systems: Design and Testing Practices*, eds. by Zoltan Papp, George Exarchakos, ISBN 978-981-10-0714-9, (Springer, Springer Science+Business Media Singapore, March 2016)
5. B. Randell, C. Landwehr, A. Avizienis, J.-C. Laprie, I.E.E.E. Trans, Dependable Secure Comput.1, 11 (2004).
6. PT 62861 Principal Component Reliability Testing for LED-based Products, Working Groups TC 34/SC 34A/PT 62861, available at http://www.iec.ch/dyn/www/f?p=103:14:0::::FSP_ORG_ID,FSP_LANG_ID:9683,25. Last assess on 11/18/2016
7. H. Pham, *Software Reliability* (U.S. Government Printing Office, 2000), http://books.google.nl/books?id=TI0Sj6er8UEC
8. W.D. van Driel, M. Schuld, R. Wijgers, W.E.J. van Kooten, Software reliability and its interaction with hardware reliability. 2014 15th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE); 04/2014
9. A.M. Neufelder, *Ensuring Software Reliability* (Marcel Dekker, New York, 1993)

10. A.M. Neufelder, Lance Fiondella, Lou Gullo, Taz Daughtrey, IEEE 1633 Standard for Recommended Practice on Software Reliability" RAMS Conference January, 2015
11. J.A. McCall, W. Randell, J. Dunham, *Software Reliability, Measurement, and Testing,* (Rome Laboratory, RL-TR-92-52, 1992)
12. S. Ismail, *Exponential Organizations: Why New Organizations Are Ten Times Better, Faster, and Cheaper Than Yours (and What to do About It)* (Diversion Books, EAN: 9781626814233, New York, 2014)
13. W.Q. Meeker Y. Hong, Reliability Meets Big Data: Opportunities and Challenges (2013), Statistics Preprints. Paper 82. http://lib.dr.iastate.edu/stat_las_preprints/82
14. W.Q. Meeker, L.A. Escobar, *Statistical Methods for Reliability Data* (Wiley, New York, 1998)
15. M.G. Pecht, *Prognostics and Health Management of Electronics* (Wiley, Hoboken, 2008)