INTERNATIONAL DESIGN CONFERENCE – DESIGN 2022
https://doi.org/10.1017/pds.2022.200

Fault-Tolerant Function Development for Mechatronic Systems

R. Stetter [1,✉] and U. Pulm [2]

[1] University of Applied Sciences Ravensburg-Weingarten, Germany,
[2] University of Applied Sciences Hamburg, Germany
✉ ralf.stetter@hs-weingarten.de

Abstract

The main focus of this paper is the exploration of fault accommodation possibilities in the context of function development. Faults occur in complex technical systems and may lead, if no accommodation entities or processes are present, to catastrophic failure. Several entities and processes exist and are applied, but mainly on the concrete levels. Faults very often concern more than one physical domain and accommodation possibilities are present in many physical or even non-physical domains. This paper explores this specific challenge and proposes an initial collection of countermeasures.

Keywords: mechatronics, functional modelling, model-based systems engineering (MBSE), fault-tolerant design, function development

1. Introduction

In industrial practice, components of complex systems such as sensors and actuators are vulnerable to faults and may deteriorate the system's performance (Yang et al. 2021). Fault-tolerant control, i.e. passive or active control algorithms with the intention to accommodate faults, has attracted large attention in the research community. However, in these research works, the design of the system itself (not of the controller) is not the main point of interest; consequently, current research works accompany these investigations (Stetter 2020, Stetter et al. 2020). In this area, several fault accommodation possibilities are already proposed, but they mainly concern design stages with concrete product models. In recent years, the enormous importance of the functional level was emphasised (Eisenbart et al. 2016) and the special challenges of functional development were investigated (Pulm und Stetter 2021). This paper explores the fault-tolerant function development of mechatronic systems; this exploration is based on the experience of the authors and numerous talks with engineers - this research is in an explorative stage, intended to foster discussion and further investigations in the design research community. The main research question can be formulated as: *How can system design on the functional level support the accommodation of faults?* For this exploration, section 2 presents the state of the art concerning fault-tolerant design and function development. Section 3 investigates the emergence of faults in technical systems currently under development in industry and section 4 elucidates possible countermeasures. Examples for such countermeasures are explained in detail on the basis of product development problem solving cycles in industry and academia in section 5. Section 6 concludes the paper and provides an outlook towards future research

2. State of the art

This section explains the state of the art in the two areas fault-tolerant design and function development.

2.1. Fault-tolerant design

In control engineering, faults are defined as temporary deviations of certain system properties from their nominal range and it is concluded that, in complex machinery and vehicles, faults cannot be completely avoided (Blanke et al. 2016). Today, several research groups worldwide investigate possibilities to accommodate faults by means of a robust control design or active fault-tolerant control. In this scope, the accommodation of faults can be defined as the presence of characteristics or activities of the technical system, which contribute to reduce or eliminate the consequences of a fault on the system's performance or safety. Current research topics in this area include, amongst others, distributed finite-time fault estimation for cyber-physical systems (Zhu et al. 2021), consensus control for multi-agent systems (Sader et al. 2021), and fault-tolerant control for networked control systems applied to underwater vehicles (Li et al. 2021). Despite the incontestable appeal of these approaches, they do not answer the question how system design can contribute to this accommodation of faults. The possibilities for this are manifold; the most frequent example is redundancy such as in the engines of a plane. A systematic approach for this kind of design is currently explored using the notion “fault-tolerant design” (Stetter 2020). This systematic approach relies on a distinction of elementary fault accommodation entities based on the level of abstraction of the product models they are applied to. Figure 1 gives and overview of these levels of abstraction and respective fault accommodation entities.

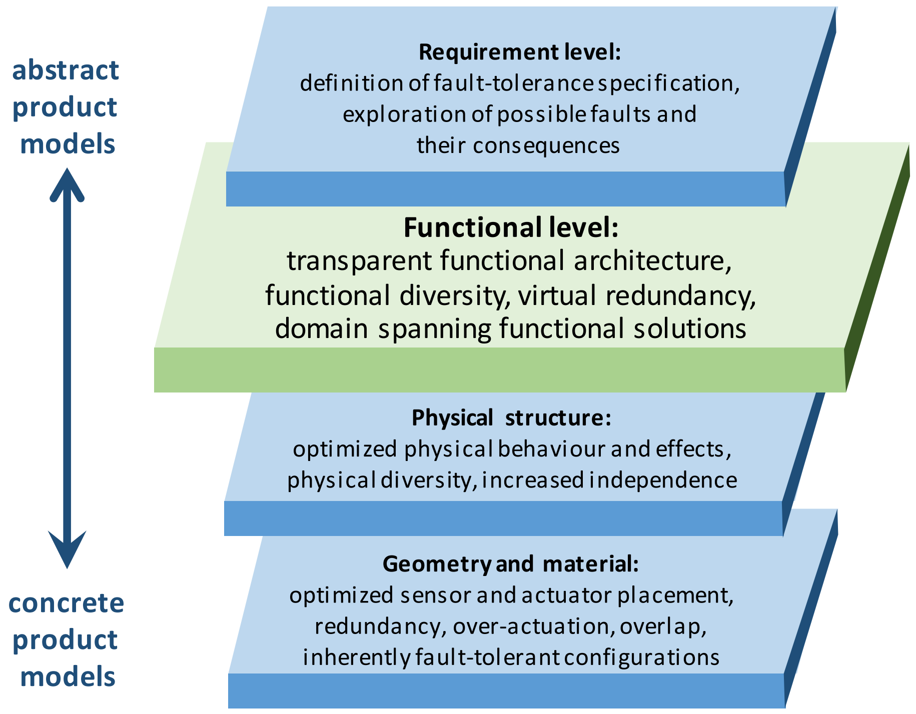

abstract product models
concrete product models
Requirement level:
definition of fault-tolerance specification, exploration of possible faults and their consequences
Functional level:
transparent functional architecture, functional diversity, virtual redundancy, domain spanning functional solutions
Physical structure:
optimized physical behaviour and effects, physical diversity, increased independence
Geometry and material:
optimized sensor and actuator placement, redundancy, over-actuation, overlap, inherently fault-tolerant configurations

Figure 1. Fault-tolerant design on different levels of product model abstraction

The most abstract level is the requirements level; the enormous importance of a conscious requirements clarification has been mentioned in numerous articles (e.g. Holder et al. 2017). On this abstract level, the most promising attempt for fault-tolerant design is the exploration of possible faults and their consequences. In control engineering, a clear distinction between fault and failure is proposed. As stated above, a fault is (only) a deviation from nominal conditions, while a failure is a possible catastrophic event, which will prevent that a system fulfils its intended function (Blanke et al. 2016). For the exploration of possible faults, it is advisable to investigate both possible failures and faults by using techniques such as fault tree analysis (FTA), failure mode and effects analysis (FMEA), and event tree analysis (ETA). The most concrete level is the geometry and material level. On this level, a large collection of rather concrete analysis and synthesis possibilities may be applied. Several powerful methods for optimised sensor and actuator placement were developed in the last years (e. g. Wrobel and Meurer 2021). Current work also explores early robust design (Goetz et al. 2021); this publication also contains a concise review of earlier work. Furthermore, fault-tolerant design can be realised by means of a conscious application of redundancy, over-actuation, overlap, and inherently fault-tolerant

design (Stetter 2020). A smaller amount of research is focused on fault accommodation on the physical level. Currently, an increasing interest on understanding and documenting the physical relationships in a technical system can be observed. Even without an explicit intention to increase fault-tolerance, many system designers realise physical diversity, i.e. they use actuators and sensors based on different physical effects within one system; in aircraft design this is even mandatory.
Even less research work has been addressing fault accommodation on the functional level. As pointed out earlier, this is the main focus of this paper with the main points being functional redundancy (both in parallel and on different levels of abstraction), tolerant and adapting functions, and an elaborate testing of functionality; these points will be addressed in the later sections of this paper.

2.2. Function development

The main intention of function development (frequently carried out by a system architecture team) is to create - starting from customer functions - more detailed functional descriptions of mechanical and electrical components as well as software functions (which in the following can be transferred to software code manually or automatically) regarding all boundary conditions (Pulm and Stetter 2021). The function development also helps integrating systems both for project management and for verifying them.
In industrial companies, function development is in integral part of the systems engineering. In the fall of 2020, the guideline VDI/VDE 2206 was republished as a draft and the name was changed to "Development of cyber-physical mechatronic systems (CPMS)". The intention of this guideline is an assistance for the essential activities in the product development of cyber-physical systems. The guideline describes the logical relationships in form of an inherent flow logic. The central representation is an updated and extended V-model (Gräßler and Hentze 2020). The main intention of its rework is the adaptation to contemporary alterations in industrial environments, which are a consequence of increasing cyber-physical system content and enhanced and expanded computer-aided synthesis and analysis methods and tools. Research groups around the globe are currently expanding the investigations in this topic area and focus on the study of requirements change risk (Gräßler et al. 2020), on automated parameter selection (Lu et al. 2021), and on domain-specific modelling approaches (Shaked and Reich 2021). Important research activities concentrate on Model Based System Engineering (MBSE - Darpel et al. 2020, Laing et al. 2020, Pottebaum and Gräßler 2020). An innovative approach is the application of graph-based design languages (GBDLs - compare e.g. Holder et al. 2017). The essential elements of industrial design processes of mechatronic and cyber-physical systems are shown in Figure 2 (based on VDI/VDE 2020 - focus on function development).

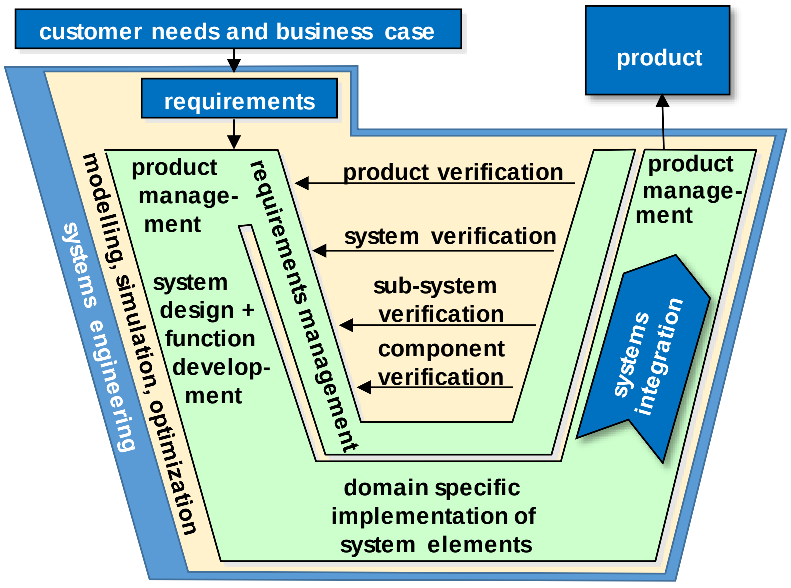


Figure 2. Essential elements of design processes of mechatronic systems

The represented V-model emphasises the large role of systems engineering and places the function development in the context of system design. Function development is logically precedent to the

domain specific implementation of system elements and includes physical and non-physical considerations. Visible is also a function "product management" which is currently present in most companies and translates customer needs and expectations as well a marketing issues into the engineering context. In the centre of the V, several verification levels are represented. Obviously, product faults can be detected in these verifications, which may be carried out rather early in a product development process. The detected faults frequently emerge because of certain conditions. This emergence of faults is the main topic of the subsequent section.
While the V-model is an appropriate representation of the overall process and the systematic development (structuring, integration, and testing) of a system, it hardly answers how to assign functions to a certain domain. A set of criteria including cost, weight, time, reliability, accuracy, endurance, performance, and fault-tolerance itself might help finding the optimal domain where to realise the function or react to a possible or actual fault (Pulm and Stetter 2021).

3. Emergence of faults

One may ask why faults still emerge in products which are results of state-of-the-art design processes. An example for such a fault might be the catalytic converter of a motor-bike. Insufficient petrol quality may cause wear of this component. The lambda-sensor placed after the catalytic converter would detect this wear and an adaptation of the motor control might accommodate this fault. Still, a fault has emerged as a result of an unexpected environmental condition. This section seeks to explore such causes of faults.
The industrial experience of the authors as well as numerous engineering bachelor and master theses led to the insight that many problems in today's product development do not occur because of direct faults of single components, but because of an unfavourable interplay of parameter values, unfavourable topologies of product variants, and unexpected side effects of certain product configurations and operation profiles. In current product development, powerful processes, methods, and tools exist to detect and accommodate direct faults of sensors and actuators. Still, in certain areas, faults occur. In mechanical engineering, these faults are often connected with production tolerances, unfavourable tolerance chain situations, non-linear effects such as friction with stick-slip, and mechanical wear of functional surfaces. In electronic sub-systems, faults, which are not detected in the initial design, often result from leakage currents, loss currents, magnetism, electro-magnetism, irradiations and changing resistances e.g. as a consequence of wear. In software, faults which are difficult to detect, commonly result from an insufficient exploration of possible parameter ranges, double or wrong parameter assignments, as well as synchronisation issues, and unpredicted effects in the interplay of different software components. Further faults that are difficult to detect result from unsatisfactory design (in most cases insufficient dimensioning of crucial components) and from unexpected use or even misuse. Additionally, unexpected environment conditions such as the presence of corrosive media and other unexpected influences such as vibrations or thermal issues may cause faults. It is important to note that, in several cases, an event such as a product malfunction is not caused by a single fault in a single domain, but by the concurrent occurrence of more than one small fault in different areas, which one their own my not even be detected by the end user. The different causes for the emergence of faults are summarised in Figure 3. It is important to note that this taxonomy may be expanded in further research.
In future product development systems, it might be possible to detect unfavourable combinations by means of artificial intelligence, if all product data and information about the underlying engineering processes is collected and certain patterns are sought, which have led to problems in the past. This can be a rich field for future research, but today a large share of the information generated in a product development process is not accessible, e.g. they are never documented or stored in proprietary data formats. Relying on the information which is available today, it is still possible to identify strategies to find countermeasures for these kind of faults, these are discussed in the subsequent section.

https://doi.org/10.1017/pds.2022.200 Published online by Cambridge University Press

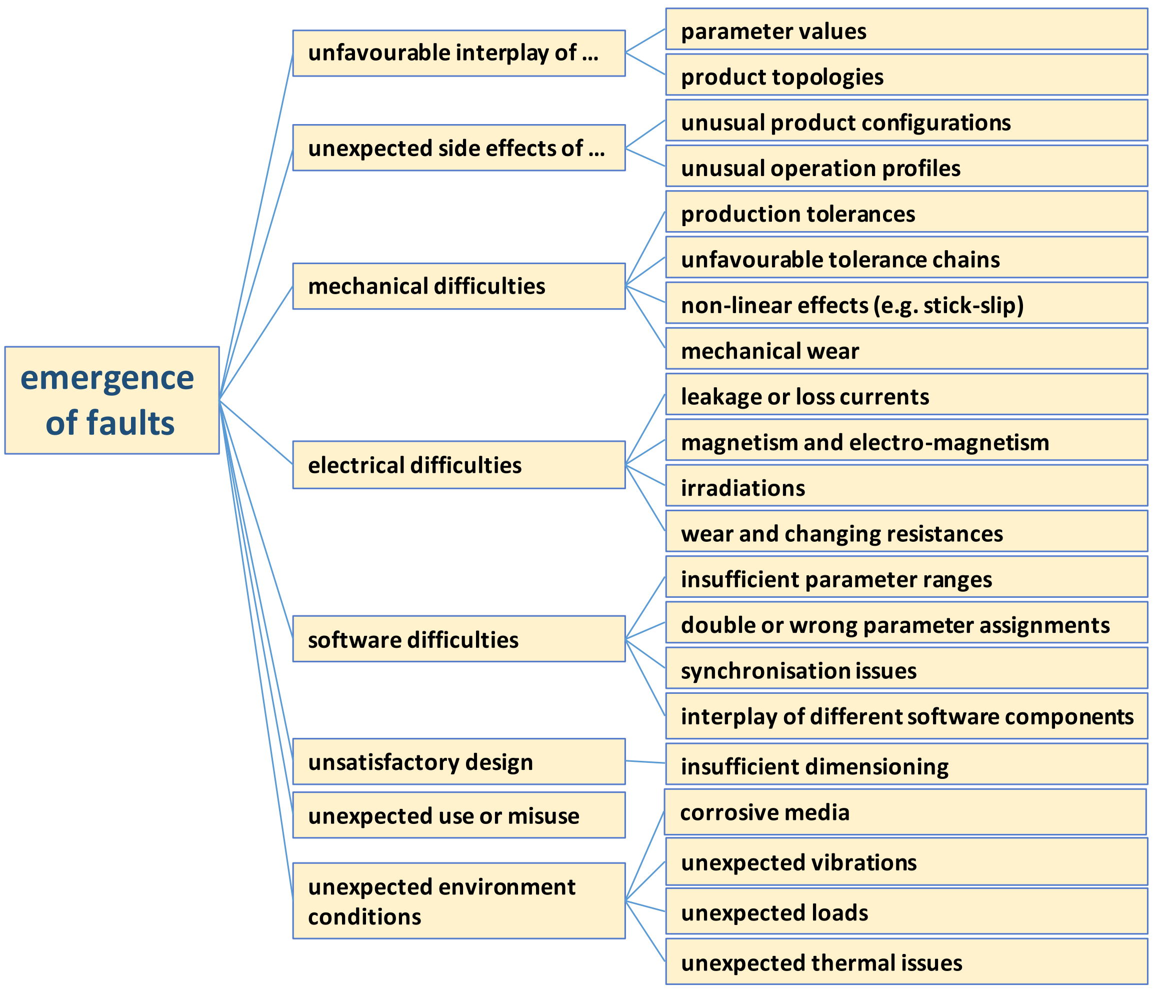

emergence of faults
unfavourable interplay of ...
parameter values
product topologies
unexpected side effects of ...
unusual product configurations
unusual operation profiles
mechanical difficulties
production tolerances
unfavourable tolerance chains
non-linear effects (e.g. stick-slip)
mechanical wear
electrical difficulties
leakage or loss currents
magnetism and electro-magnetism
irradiations
wear and changing resistances
software difficulties
insufficient parameter ranges
double or wrong parameter assignments
synchronisation issues
interplay of different software components
unsatisfactory design
insufficient dimensioning
unexpected use or misuse
corrosive media
unexpected environment conditions
unexpected vibrations
unexpected loads
unexpected thermal issues

Figure 3. Causes for the emergence of faults

4. Countermeasures for faults

In general, countermeasures for faults, i.e. entities or processes to accommodate faults are possible on the geometry and material level, the physical level, as well as the functional level (on the requirements level only objectives ca be formulated but no countermeasures should be described in order to remain "solution-neutral"). The focus of this paper is the most abstract level for countermeasures - how can a conscious function development be applied to counteract faults which occur in the multi-domain interplay of the components of complex systems. It is important to point out that these countermeasures are not limited to a certain physical domain, but may include solutions involving different physical domains (mechanical, electrical, etc.) and even solutions in the fields of controlled product operation, user involvement and warnings, as well as legislation, and supply chain management issues. If a certain operation condition is detected as harmful for the system, the control systems of the product my limit the time in this certain condition (e.g. a time-limited over-boost) or if a system fails to be in accordance with certain emission laws, a certain operation condition can be prevented completely. Solution concepts for countermeasures can be found in all domains and are frequently a combination of mechanical, electrical, and software elements; the solution elements may stretch beyond a pure technological solution (Pulm and Stetter 2021). The main challenge is that design engineers need to be aware about the potential of multi-domain solution components and that they need to be able to apply them in a systemic manner, which means that they need to be aware of the multitude of consequences in a complex system and to realise a systematic application. It is rather obvious that design engineers need to understand the interconnected functionality of subsystems in their product. A profound knowledge of functional analysis methods (such as the integrated function modelling framework (IFM - Eisenbart et al. 2016)) are one prerequisite for identifying fault countermeasures in a systemic manner. In the

https://doi.org/10.1017/pds.2022.200 Published online by Cambridge University Press

function analysis of faults, their consequences and possibilities to accommodate them, control and diagnosis functionalities deserve special attention. Especially helpful can be the application of a relation-oriented function modelling technique, which distinguishes between useful and harmful function, such as one of the function modelling methods proposed in the TIPS/TRIZ (theory of inventive problem solving) community (compare e.g. Banciu et al. 2011). The syntax of function modelling for this specific modelling method is shown in Figure 4 together with a simplified example based on one of the product development examples in Section 5 (balanced two-wheel scooter).

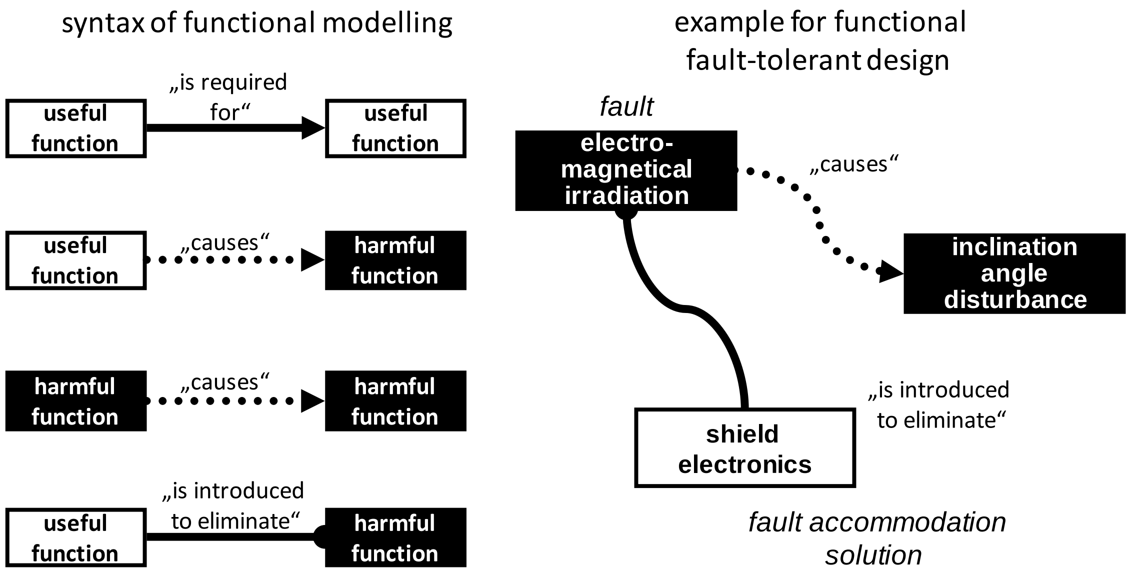

syntax of functional modelling
useful function
„is required for“
useful function
useful function
„causes“
harmful function
harmful function
„causes“
harmful function
useful function
„is introduced to eliminate“
harmful function
example for functional fault-tolerant design
fault
electro-magnetical irradiation
„causes“
inclination angle disturbance
shield electronics
„is introduced to eliminate“
fault accommodation solution

Figure 4. Accommodation of the fault "inclination angle disturbance"

It is clearly visible in this kind of function model that the fault "electro-magnetical irradiation" is a harmful function which may cause another harmful function "inclination angle disturbance". It is possible to document that the fault accommodation solution "shield electronics" is introduced as a useful function to eliminate the effects of this fault.

Design engineers who want to realise a holistic accommodation of faults need to know about solution possibilities in many domains. They may be assisted by an abstract database of solution elements with a description of certain functional characteristics and by methods to model functional redundancies (Pulm und Stetter 2021).

In many cases, several possible countermeasures in different domains may be available. The design engineers need to be able to analyse these different countermeasures and to establish a domain-spanning evaluation including criteria such as

- degree of fault accommodation (how good are the consequences of the fault prevented?),
- fault range (will the consequences of a slightly different fault also be prevented?),
- user information (will the user be informed concerning the emergence of the fault?),
- legal information (will diagnosis systems, which are legally required, capture the emergence of the fault?),
- additional system cost and weight, etc.

A concise systematic guideline how entities or processes to accommodate faults may be developed is still the object on on-going research. However, some concrete measures in a systematic context can already be listed in this paper (these measures are not limited to the functional level):

- Redundancy: The most straight forward measure is the integration of redundant system elements such as additional sensors, actuators and communication lines which can replace missing elements in the case of a fault.
- Monitoring and diagnosis on different levels: Many complex systems have some kind of diagnosis system monitoring the state and the functioning of the main system itself. In the case of a fault,

default values, fall-back functions, limitations to the performance, or a warning for the use might be applied.

- Tolerance analysis: Tolerance analysis essentially means to combine the dimensions and tolerances of assembly components and their assembly sequence in an analytical model that allows to determine the tolerance ranges of certain product aspects which are important for the product assembly feasibility or product function (compare e.g. Corrado et al. 2016). As some faults are consequences of tolerance chains, these can be investigated using tolerance analysis and accommodation possibilities can be found.
- Robust design: Generally robust design aims to create products insensible to uncontrollable variations and is based on the identification of an optimal parameter set which includes a small variance of the target values (product performance objectives) as a constraint (Kemmler et al. 2015). As some smaller faults might have similar effects as those variations, robust design can also be understood as one measure for fault-tolerant design.
- Testing: An elaborate testing program including the inclusion of probable fault conditions can be one cornerstone in the development of fault-tolerant products.
- Constraints: One possibility for limiting the effect of faults is the employment of "hard-programmed" constraints for certain parameters, i.e. fixed limits for certain actuators, which can be accompanied by additional mechanical limit stops. Also within the design process, constraints for what methods might be used can be applied (design constraints).
- Applying the proposed measures of design for safety: the measures described in the design for safety guidelines such as inherent safe design or danger warnings seek to minimise to risk of injury or malfunctions and can consequently also be applied in the scope of fault-tolerant design.

Besides these rather concrete measures which may help to increase certain aspects of the fault-tolerance of technical systems, a conscious function development of fault accommodation entities is inevitable, because the measures on the concrete levels described above cannot counteract principal system weakness in terms of fault-tolerance. Sensible is a conscious development of fault accommodation functions in the system design and function development stage (compare Figure 2). On a functional level, faults can be understood as harmful functions (compare Figure 4) and useful functions need to be defined in order to counteract ("eliminate" in Figure 4) them. Their definition is essentially comparable to the definition of "normal" product functions (compare Section 2).

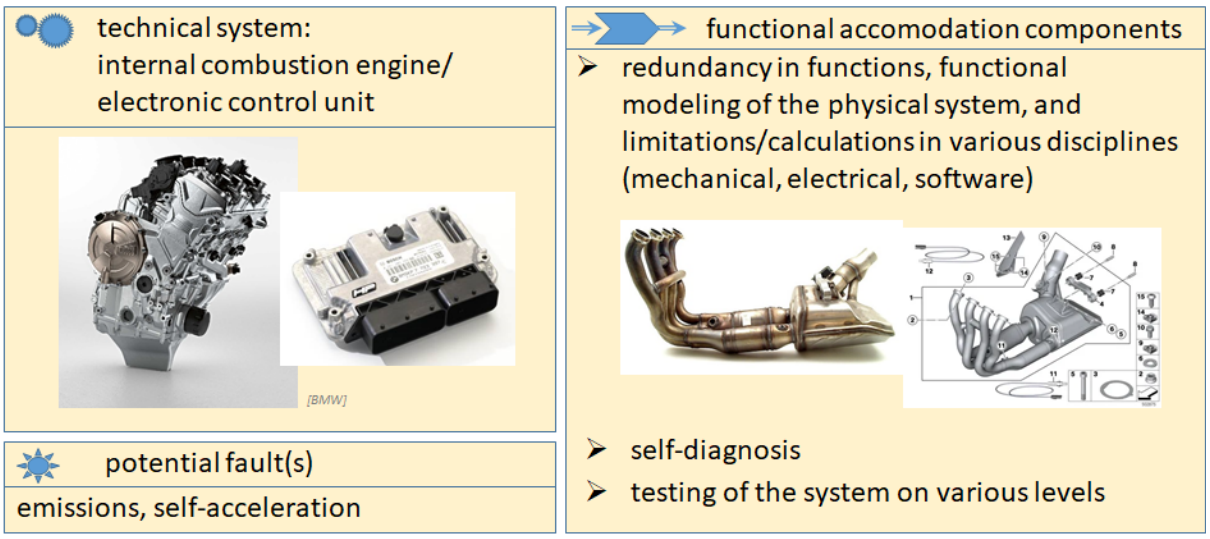
5. Product development examples

A good example for a complex mechatronic product offering a vast number of possibilities for faults and thus incorporating many approaches for fault-tolerant design is a modern internal combustion engine with its electronic control unit (Figure 5). Faults may occur on the mechanic level (e.g. tolerances and wear), the electronic level (e.g. malfunction of sensors), or the software level (e.g. a too sensitive calibration of functions or the interplay of various functions) - and of course due to a mixture of these aspects. The consequences of these faults might affect driver safety ("self-acceleration") and emissions as the most important ones.

The main control of the engine is based on a redundant functional modelling of the physical system, i.e. the mechanical engine is represented within the control unit via two distinguished physical models, which calculate the output of the engine, the torque, separately - and in case of discrepancies fall back to a safe mode. The output of a sensor is validated by roughly calculating the value via other sensor output - and again setting a default value in the case of a large aberration. In the same way, actuators are limited by mechanical limitations as well as software limits. A main sensor such as the electronic throttle is realised via two separate and parallel electronic sensors. Most sensors and actuators are adapted automatically when starting up the engine in order to compensate production tolerances. In the case of emissions, there are various control loops ensuring a minimum of emissions. While the engines combustion itself can be very clean, a catalyst serves as a buffer reducing start-up and dynamic effects. A perfectly clean combustion is guaranteed by various oxygen sensors controlling the combustion, the deterioration of the catalyst, and the functioning of the sensors themselves via intelligent diagnosis functions.

Just as diagnosis and safety functions might sum up to more than half of the control unit functions, testing is the main task in the development of an engine. This happens - as proposed in the V-model - on different

levels of the products structure including (automated) testing of software and the electronic system itself with software-in-the-loop (SIL) and hardware-in-the-loop (HIL), various test benches, as well as real life testing in various environmental conditions.
There are also design constraints in the software programming such as not using complex functions (such as pointers), using model-based software engineering (MBSE) and automated program coding.
As described above, engineers have to decide in which domain a function has to be realised. In the same way, a possible or actual fault can be accommodated in different domains, too. One specific example might be the safe shifting of a gear. This operation might by faulty and the gear might not switch properly due to production tolerances, wear, or a weak actuation by the driver. It can be answered by a mechanical solution (more precise gears), by additional sensors or actuators, or by software functions changing the engine torque in a supporting way, which has been the most stable solution.


technical system:
internal combustion engine/
electronic control unit

[BMW]

potential fault(s)
emissions, self-acceleration

functional accomodation components
- redundancy in functions, functional modeling of the physical system, and limitations/calculations in various disciplines (mechanical, electrical, software)
- self-diagnosis
- testing of the system on various levels

Figure 5. Accommodation of possible faults in an engine

Another example product development concerns balanced two-wheel scooter, which is one of the use cases of a research project investigating GBDLs (ICED 2019). One main challenge in the operation of this kind of vehicle is to maintain a desired inclination angle. In theory, sensors, which are currently available on the market, can accurately measure this inclination angle. It is also in general possible to choose drive motors which possess the dynamic performance to act as actuators and achieve the intended angles. Consequently, it should be possible to synthesise an effective control algorithm; commercial system with well-functioning control algorithms are available on the market. Still, in the development project it was a continuous challenge to control and maintain this inclination angle. Domain specific countermeasure possibilities would be better sensors and better actuators. A detailed analysis lead to the insight, that the problem was partly caused by irradiations and that a shielding of the electronics and accelerometers was an effective countermeasure. In this case, it was realised using a 3-D printed housing for these components and a shield by means of an aluminium adhesive tape (Schuster and Pahn 2018) as shown in Figure 6. The problem was also partly caused by fluctuations in the sensor output, here the application of appropriate filters was improving the system behaviour.

6. Conclusions and Outlook

The main intention of this paper was to explain and demonstrate how a conscious function development of mechatronic systems may expand and enhance the capability of these systems to accommodate faults, i.e. how a conscious design and selection of characteristics and systems for certain control and diagnosis activities can contribute to a reduction of the consequences of unavoidable faults.

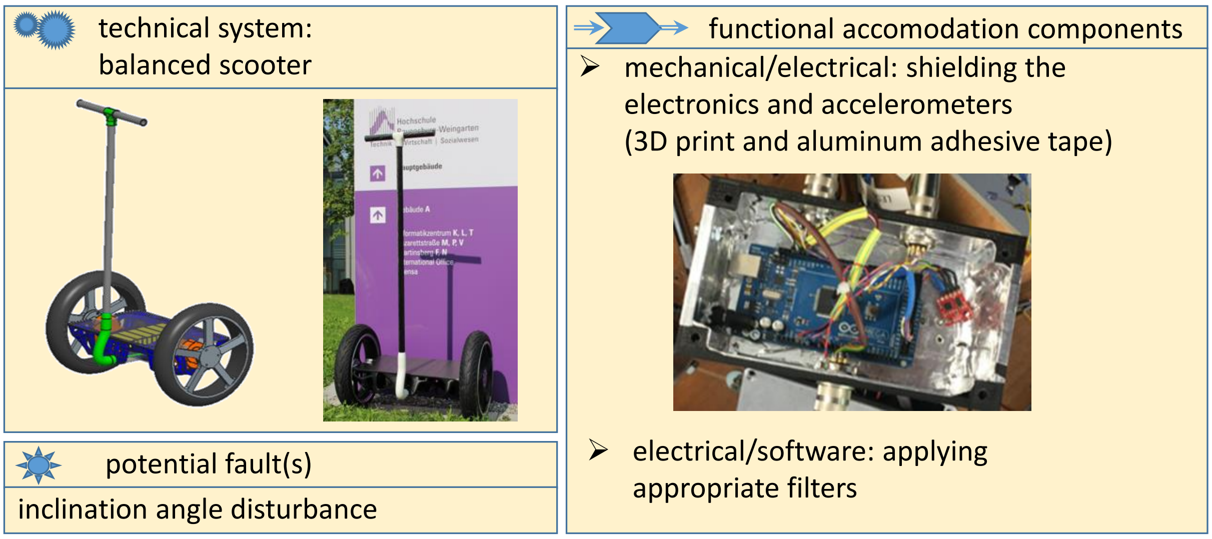

technical system: balanced scooter
potential fault(s)
inclination angle disturbance
functional accomodation components
mechanical/electrical: shielding the electronics and accelerometers (3D print and aluminum adhesive tape)
electrical/software: applying appropriate filters

Figure 6. Accommodation of the fault "inclination angle disturbance"

Main challenges are the multiple potential causes for the emergence of faults - most of them go beyond the nominal operation of isolated components. Another challenge is that possible countermeasures can be possible in all involved technical domains and may even include non-technical aspects such as legal aspects. Frequently, promising solutions will combine solution elements in more than one domain. An initial list of principles for the function development of accommodation characteristics and activities could be compiled. The explanation was based on experience in product development examples; the main source of insight are several case studies. It is important to note that this research is still in an exploratory stage. However, the authors believe that this kind of observation and discussion is valuable for the research community. An expansion of the underlying theory concerning fault-tolerant design and the function development of mechatronics systems is planned.

Acknowledgement

A part of the research work was carried out in the scope of the project "Automatisierter Entwurf eines geometrischen und kinetischen digitalen Zwillings einer Rohbaufertigungsanlage für die Virtuelle Inbetriebnahme (TWIN)" which is funded by the German Federal Ministry of Education and Research.

References

Banciu, F.; Drăghici, G.; Pămîntaş, E. and Grozav I.: Identify Functions, Failure Modes, Causes and Effects Using the Triz Functional Modeling. Proceedings of the International Conference on Manufacturing Science and Education, (MSE 2011), Sibiu, 2011.

Blanke, M.; Kinnaert, M.; Lunze, J.; Staroswiecki, M. Diagnosis and Fault-Tolerant Control; Springer: New York, NY, USA, 2016.

Corrado, A.; Polini, W.; Moroni, G.; Petrò, S.: 3D Tolerance Analysis with Manufacturing Signature and Operating Conditions, Procedia CIRP, Volume 43, 2016, pp. 130-135, ISSN 2212-8271, https://doi.org/10.1016/j.procir.2016.02.097.

Darpel, S.; Beckman, S.; Ferlin, T.; Havenhill, M. Parrot, E.; Harcula, K.: Method for tracking and communicating aggregate risk through the use of model-based systems engineering (MBSE) tools. Journal of Space Safety Engineering 7 (2020), pp. 11–17, https://doi.org/10.1016/j.jsse.2020.01.001.

Eisenbart, B., Gericke, K., Blessing, L. and McAloone, T (2016) "A DSM-based Framework for Integrated Function Modelling: Concept, Application and Evaluation", Research in Engineering Design, 2016, Vol. 28 No. 1, pp. 25-51. https://doi.org/10.1007/s00163-016-0228-1.

Goetz, S.; Roth, M., Schleich, B.: Early Robust Design—Its Effect on Parameter and Tolerance Optimization. Applied Sciences 2021, 11, 9407, https://doi.org/10.3390/app11209407.

Gräßler, I.; Hentze, J.: The new V-Model of VDI 2206 and its validation. at – Automatisierungstechnik 2020; 68(5): pp. 312–324, https://doi.org/10.1515/auto-2020-0015.

Gräßler, I.; Ole, C.; Scholle, P.: Method for Systematic Assessment of Requirement Change Risk in Industrial Practice. Applied Sciences, 2020, 10, 8697; doi:10.3390/app10238697.

Holder, K., Zech, A., Ramsaier, M., Stetter, R., Niedermeier, H.-P., Rudolph, S. and Till, M. (2017) "Model-Based Requirements Management in Gear Systems Design based on Graph-Based Design Languages". Appl. Sci. 2017, 7, 1112, https://doi.org/10.3390/app7111112.

Kemmler, S.; Fuchs, A.; Leopold, T.; Bertsche, B.: Comparison of Taguchi Method and Robust Design Optimization (RDO) - by application of a functional adaptive simulation model for the robust product-optimization of an adjuster unit. In: Proceedings of the 12th Weimar Optimization and Stochastic Days, 2015, http://dx.doi.org/10.18419/opus-8435.

Laing, C.; David, P.; Blanco, E.; Dorel, X.: Questioning integration of verification in model-based systems engineering: an industrial perspective. Computers in Industry 114 (2020) 103163, https://doi.org/10.1016/j.compind.2019.103163.

Li, H.; Pan, J.; Zhang, X.; You, J.: Integral-based event-triggered fault estimation and impulsive fault-tolerant control for networked control systems applied to underwater vehicles. Neurocomputing 442 (2021) 36-47, https://doi.org/10.1016/j.neucom.2021.02.035.

Lu, J.; Chen, D.; Wang, G.; Kiritsis, D.; Törngren, M.: Model-Based Systems Engineering Tool-Chain for Automated Parameter Value Selection. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2021, accepted for inclusion.

Pfeifer, S.; Seidenberg, T.; Jürgenhake, C.; Anacker, H.; Dumitrescua, R.: Towards a modular product architecture for electric ferries using Model-Based Systems Engineering. Procedia Manufacturing 52 (2020) pp. 228–233, https://doi.org/10.1016/j.promfg.2020.11.039.

Pottebaum, J.; Gräßler, I.: Informationsqualität in der Produktentwicklung: Modellbasiertes Systems Engineering mit expliziter Berücksichtigung von Unsicherheit. Konstruktion 72 (11-12) 2020, 76–83.

Pulm, U.; Stetter, R.: Systemic mechatronic function development. In: ICED 2021, Proceedings of the Design Society, 1, pp. 2931-2940. doi:10.1017/pds.2021.554.

Rudolph, S. Übertragung von Ähnlichkeitsbegriffen. Habilitationsschrift, Fakultät Luft- und Raumfahrttechnik und Geodäsie. Habilitation Thesis, Universität Stuttgart, Stuttgart, Germany, 2002.

Sader, M.; Chen, Z.; Liu, Z.; Deng, C.: Distributed robust fault-tolerant consensus control for a class of nonlinear multi-agent systems with intermittent communications. Applied Mathematics and Computation 403 (2021) 126166, https://doi.org/10.1016/j.amc.2021.126166.

Schuster, J. and Pahn, F. (2018) Entwicklung und Bau zweier konzeptionell unterschiedlicher Segways. Bachelor-Thesis Ravensburg-Weingarten University (RWU).

Shaked, A.; Reich, Y.: Using Domain-Specific Models to Facilitate Model-Based Systems-Engineering: Development Process Design Modeling with OPM and PROVE. Applied Sciences 2021,11, 1532. https://doi.org/10.3390/app1104153

Stetter, R.: Fault-Tolerant Design and Control of Automated Vehicles and Processes. Insights for the Synthesis of Intelligent Systems. Springer, 2020, https://doi.org/10.1007/978-3-030-12846-3.

Stetter, R.: Approaches for Modelling the Physical Behavior of Technical Systems on the Example of Wind Turbines. Energies (2020), Vol. 13, No. 8, 2087, https://doi.org/10.3390/en13082087.

Stetter, R.; Göser, R.; Gresser, S.; Till, M.; Witczak M.: Fault-tolerant design for increasing the reliability of an autonomous driving gear shifting system. Maintenance and Reliability, Vol. 22, No. 3, 2020, pp. 482 – 492. http://dx.doi.org/10.17531/ein2020.3.11.

VDI/VDE 2206 – Entwurf: Entwicklung cyber-physischer mechatronischer Systeme (CPMS). Beuth: 2020.

Walden, D. D.; Roedler, G. J.; Forsberg, K.; Hamelin, R. D.; Shortell, T. M.: Systems engineering handbook: A guide for system life cycle processes and activities, 4th ed. Wiley, 2015.

Wrobel, M.; Meurer, T.: Optimal Sensor Placement for Temperature Control in a Deep Drawing Tool, IFAC-PapersOnLine, Volume 54, Issue 11, 2021, pp. 91-96, https://doi.org/10.1016/j.ifacol.2021.10.056.

Yang, X.; Li, T.; Wu, Y.; Wang, Y.; Long, Y.: Fault estimation and fault tolerant control for discrete-time nonlinear systems with perturbation by a mixed design scheme. Journal of the Franklin Institute 358 (2021), pp. 1860 - 1887, https://doi.org/10.1016/j.jfranklin.2020.12.024.

Zhu, B, Wang, Y, Zhang, H.; Xie, X.: Distributed finite-time fault estimation and fault-tolerant control for cyber-physical systems with matched uncertainties. Applied Mathematics and Computation 403 (2021) 126195, https://doi.org/10.1016/j.amc.2021.126195.