Fraunhofer IKS

FRAUNHOFER INSTITUTE FOR COGNITIVE SYSTEMS IKS

JULY 30, 2020

FLEXILIENT END-TO-END ARCHITECTURES

FLEXIBILITY, INTELLIGENCE AND RESILIENCE FOR DEPENDABLE CLOUD-BASED CYBER-PHYSICAL SYSTEMS

AUTHORS: CHRISTIAN DRABEK, ANNA KOSMALSKA

ABSTRACT

Highly intelligent, massively connected, autonomous systems featuring state-of-the-art technologies offer as many opportunities as challenges. The Internet-of-Things paradigm enters all areas of life. However, it is not enough to just provide intelligence or autonomy to systems. They must be able to connect to other systems, provide services and end-to-end communications, adapt to changing needs and be dependable at the same time.

The challenge is resilience — the persistence of dependability when facing changes — and how it can be defined in the context of end-to-end architectures, which consist of many layers of components, both software and hardware, which in turn can have different safety, availability and dependability requirements. We propose the term *flexilience* — the combination of flexibility, intelligence and resilience to describe this area of conflict. Flexibility thrives to constantly and perfectly adapt to the present conditions, while intelligence continuously increases the systems' cognitive capabilities and resilience ensures its dependability in changing conditions. Therefore, flexilience is persistent dependability and optimized performance in cognitive systems when facing changes. In intelligent autonomous systems, considering only worst-case scenarios during the design phase would result in dramatically limited performance. For example, potential cloud or edge services in such scenarios could not be used for any safety-related functions. The actual situation and risk should thus be taken into account.

In this paper, we present a novel approach for designing and managing such systems at runtime that allows safety aspects to be evaluated and guaranteed not only during the design phase and for worst-case scenarios, but also at runtime in line with the current situation. As a result, we can move functions, including those that are safety related, to the cloud or edge for improved performance.

1 INTRODUCTION

In the future, the world will rely on widespread, massively connected **cognitive systems**. Growing market and customer needs are pushing research towards dependable cloud-based cyber-physical systems. The challenge is to design them to simultaneously provide **resilience**, **intelligence** and **flexibility**.

Technology affects all areas of our lives. Systems are expected to be intelligent, autonomous, safe and adaptable. However, the current state of technology does not correspond to these needs. By itself, an autonomous, intelligent, mobile system is limited due to storage, energy and processing constraints, thus leading to user frustration and either a lack of desired functionalities, or an insufficient level of safety.
Such systems exist in **changing environments** and interact with other systems and humans. For **safety-critical functions**, **maximizing performance** is often neglected in order to guarantee safety. This results in system designs with limited functionality. However, the growing complexity of the world is creating a need for systems that can adapt and modify their behavior to address new situations. Complex systems and the increasing demands of the world call for adaptive and adaptable systems that can change their configuration — and thus their behavior — to meet new requirements or deal with a new situation.

A wide range of industries can benefit from such systems or rather **system-of-systems**, from health care to manufacturing. A prominent example is the automotive industry in the design and manufacture of self-driving vehicles. Connected autonomous vehicles can offer a much higher level of optimization and demonstrate completely new opportunities. They can help to avoid accidents [1] and can maintain safe operation while increasing performance. Such systems-of-systems need to be safe and reliable to ensure proper handling of hazardous situations and to provide uninterrupted services and an excellent user experience. In general, an architecture comprises "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution" [2]. For systems-of-systems, an **end-to-end architecture** also takes into account the availability and reliability of the interacting systems.
Flexible, resilient and intelligent end-to-end architectures are needed to utilize various services that offer information and resources and degrade gracefully if they are no longer available, such as if the (wireless) connection fails. They should be dynamic and cover end-to-end communications — for example, from a mobile endpoint, like an autonomous vehicle or robot, to edge infrastructures such as road cameras or facility sensors, to cloud services. While a holistic approach is required to properly address the multivariate problems, a system-of-systems is typically not implemented by a single entity. We are proposing a development approach that systematically captures these relationships between subsystems and thereby makes the relevant properties of each subsystem explicit. This allows separation of subsystems and enables distributed development of the system-of-systems.
This white paper is structured as follows: Chapter 2 discusses the challenges of dependable cloud-based cyber-physical systems(-of-systems). They should combine aspects of resilience, intelligence and flexibility, for which we use the term ***flexilience*** that will be introduced in Chapter 3. Chapter 4 then outlines our solutions for developing flexilient end-to-end architectures.

2 DEPENDABLE CLOUD-BASED CYBER-PHYSICAL SYSTEMS (-OF-SYSTEMS)

This chapter presents three scenarios for dependable cloud-based cyber-physical systems(-of-systems) and discusses the associated challenges.

2.1 CONNECTED AUTONOMOUS VEHICLES

Connected autonomous vehicles have outstanding potential not only for personal transportation, but also for fleets of autonomous taxis, buses or trucks, providing cheaper and safer transportation. While safety can never be compromised and operability of the system is important at all times, on-board resources such as processing power, sensor quality and battery capacity are not sufficient to enable mobility at a reasonable cost. Moreover, customers expect autonomous vehicles to deliver driving performance that is superior to what human drivers are capable of.

In many situations, autonomous vehicles must make decisions related to things like speed, maneuvering and route selection, all of which would benefit from additional resources, whether it is more information or processing power. A prominent example is the detection of pedestrians or other road users. Edge infrastructures such as high resolution cameras and the abundance of cloud processing power make it possible to create a comprehesive environment model that could allow faster vehicle speed with acceptable risk. Another potential source of information is other vehicles in the vicinity. Still, much like the vehicle may need to adapt its sensors to the current ambient conditions, it has to adjust to the availability of external resources to ensure constant safe and reliable operation. In addition, given that algorithms are expected to evolve quickly, they need to be updated on a regular basis.

A related scenario is the control and coordination of autonomous vehicles in semi-structured environments such as parking lots [3]. Automated valet parking systems will benefit from connectivity in order to exchange additional sensor data or information related to localization, available parking space and guidance for the vehicles. However, to ensure permanent safe operation, loss of connectivity must always be anticipated. At a minimum, a reliable system should be able to continue operation in a degraded mode if communication with certain vehicles is not possible. For example, it should clarify the conditions under which a vehicle can continue to operate based solely on its sensors.

2.2 BUILDING AND INFRASTRUCTURE MANAGEMENT

Another example is building management. Today's buildings are equipped with various systems that allow them to reduce energy consumption, heating costs or the carbon footprint, as well as increase safety and security among other things. Such systems can be more costeffective and easier to maintain and update if the functions are relocated to the cloud, even potentially including critical or safety-related functions. Apart from single buildings, smart cities [4] can also be managed with this approach. Similar benefits can be achieved for infrastructures such as network or traffic management systems. While such systems must remain operational even if individual components fail, they cannot always be built with full redundancy for every sensor due to cost, space or other reasons. However, some inherent redundancy does exist through the abundance of sensors, a fact that also tests the scalability of any planning tool. Intelligent countermeasures can exploit this environment by calculating the failed sensor readings from other inputs as an example. By evaluating the increased risk from using a virtual sensor, the system's performance can be adjusted to still meet safety rules. Alternatively, the system could switch to a different mode of operation that does not require the missing sensor information.

2.3 WAREHOUSES AND SORTING FACILITIES

Modern warehouses and sorting facilities at delivery companies, online stores and other facilities are seeing an increasing number of automated vehicles or machines that are designed to achieve the highest level of efficiency [5]. Mobile robots or unmanned forklifts can maximize their performance when coordinated from a central control system. Moreover, the planning could span multiple facilities to achieve just-in-time delivery and production. Safety of the individual machines, especially when operating in the same space as human workers, may not be compromised in the event of control signal loss or hardware or software failure. However, it is not always possible to simply stop the machine, which can be a costly measure. The machine could operate safely at a degraded performance level until it can be fixed or moved out of a critical area. If a machine has to be shut down for maintenance, the remaining facility has to remain operational.

2.4 MAIN CHALLENGES

The following summarizes the main challenges of dependable, autonomous, cyber-physical systems and system-of-systems.

— Affordable **safety and efficiency** by dynamically adjusting the system's performance to an acceptable risk depending on the current situation and anticipating countermeasures that improve reliability.

— Predictable **real-time behavior** with sufficiently low-latency and jitter for control stability across the end-to-end architecture.

— Interoperable **connectivity** for coordination and information exchange between various (sub)systems created by different vendors

— Hardened **security** by identification and mitigation of additional threats to the system's safety through expansion of the system boundaries.

— Seamless **data uploads**, **updates** and **maintainability** to collect training data and distribute changes without interfering with normal operation.

2.4.1 AFFORDABLE SAFETY AND EFFICIENCY

Cyber-physical systems interact with their environments and with humans in some instances. Even in faulty states, exceptional environmental conditions or situations with unexpected human behavior, these systems must remain safe. The meaning of safe depends on the specific domain with boundaries defined by government legislation and standards. For example, ISO3691-4 [6] discusses the safety requirements and verification of driverless industrial trucks and automated guided carts. In the automotive domain, safe design to counteract hardware failure is performed according to the functional safety standard ISO26262 [7] and safety design to counteract performance limitation is performed according to ISO/PAS 21448 [8]. This normally includes the analysis of hazardous situations that the system might encounter and assessing the resulting risks that comprise exposure, severity and controllability. Thorough documentation must demonstrate that validated strategies and countermeasures ensure an acceptable level of residual risk. This typically results in stricter requirements for autonomous vehicles on public roads and less strict rules for small mobile robots operating in structured environments without human intervention. A safety decomposition can break the safety goals down to requirements at the component level.

Safety may never be compromised [9]. Ensuring safety by impeding reliability or availability reduces costs during development, but will be more expensive in the field. Utilizing dissimilar redundant components, such as in aircraft, is neither cost-effective nor always feasible. Similarly, running all safety-related functions locally, as in traditional safety systems, would limit the performance. Such systems will not fully benefit from the potential and resources that cloud or edge services offer, which requires new ways to optimize the system without violating any safety goals.

2.4.2 PREDICTABLE REAL-TIME BEHAVIOUR

Smooth, controlled motion requires data and control signals to arrive at the right time. Moreover, reactions to faults must happen before hazards can surface. This is expressed with the fault tolerant time interval, which should be longer than the time needed to detect the fault and switch to a safe state. The nature of cyber-physical systems usually necessitates a transition time that must be handled gracefully. Especially in the case of end-to-end architectures where multiple components are involved, the timing must be predictable to ensure reactions before the defined deadlines — in other words real-time behavior. This includes processing and transmission delays — such as by available processing power and bandwidth as well as jitter caused by different cycle times or scheduling mechanisms.

While several methods exist [10] to solve such multivariate problems, moving functions into the cloud inevitably introduces the possibility that the cloud function becomes unavailable. The system must therefore be able to detect this situation and enter into intermediate local operation mode in a timely manner as described in *Figure 1*. Once the cloud is available again, upgrading the function requires a second transition that has to be taken into consideration.

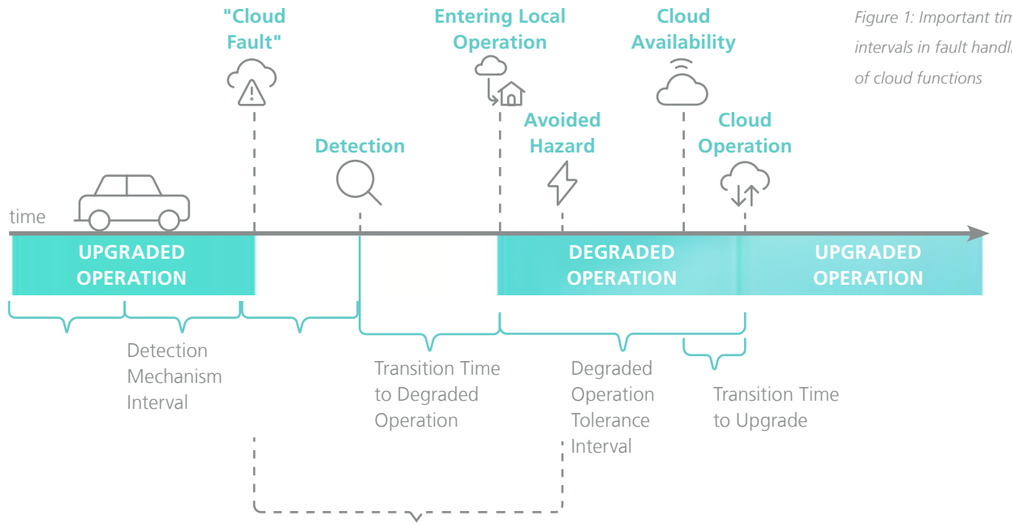


Figure 1: Important time intervals in fault handling of cloud functions

2.4.3 INTEROPERABLE CONNECTIVITY

In the context of cloud-based systems, interoperability allows the smooth management of application workloads and distribution of resources to maximize performance, avoid overloads and easily gather necessary information. Interoperability is key to connecting more and more systems developed by different vendors [11]. This compatibility requires standardized interfaces in order to dynamically negotiate interactions, such as with service contracts. Conformance must be monitored during runtime and/or certified beforehand to isolate and mitigate hazards caused by unintended behavior. Moreover, end-to-end architectures often require the integration of communication systems using different technologies. The increasing demand for connectivity requires communication at a high level of performance and reliability. It must be flexible and efficient, guarantee a high quality-of-service, and support multiple concurrent requests. Even when communication technology steadily improves, sending raw sensor data or control signals will, on a large scale, bring any system to its limits. However, extensive pre- or post-processing of signals demands additional processing power and a trade-off must be found, possibly at runtime.

2.4.4 HARDENED SECURITY

Decisions are based on available information and safety often relies on sound decisions made by the (autonomous) system. As such, if the system is vulnerable to malicious attacks [12] that provide false information, safety can be compromised. Especially in systems with varying system boundaries, there are two possible weak spots that are interesting for potential attackers. One is the communication between systems, where the exchange between vehicles, robots and cloud services can be intercepted or manipulated. The second vulnerability is a potentially malicious device at the other end of the connection. Hence, devices that participate in end-to-end architectures must somehow establish trust or validate any information received. Additional challenges include ensuring data security, privacy and authentication [11].

2.4.5 SEAMLESS DATA-UPLOADS, UPDATES AND MAINTAINABILITY

Autonomous systems should operate continuously, without interruption. On the other hand, regular software updates are essential, to improve system performance, safety as well as security and to provide new functionalities according to changing client needs, requirements and novel technologies. This is why updates must be performed dynamically, without interrupting operation of the system [13]. While dynamic software updates bring many opportunities, such as avoiding downtime, they also bring numerous risks, such as unexpected behavior of the system during an update, or uncertainty as to whether a system will remain safe and operable after an update. Furthermore, collected runtime data must be uploaded at some point to maximize the available training data, which is necessary for improving the results of machine learned components. For example, robots in a warehouse can exchange their experience with different learned algorithms to grasp items. In data-sensitive use cases, such as smart homes, this might also utilize a more privacy-friendly federated learning approach. However, it still requires a coordinated exchange of information that does not interfere with the normal operation of the system.

Learn more: Challenges with Cloud-based systems

safe-intelligence.fraunhofer.de

3 FLEXILIENCE

To overcome the main challenges of dependable cloud-based systems described in the previous section, we propose to design these systems with a focus on *flexibility*, *intelligence* and *resilience*. We do this by combining these qualities into a new term that we refer to as *flexilience*, which we view as persistent dependability and optimized performance in cognitive systems when facing changes.

Flexilience includes the ability of a system to continuously deliver the best possible service, maintain relevant safety levels and service reliability, and improve itself in unpredictable, constantly changing conditions by adapting to current context and learning from experience; in other words by being flexible, resilient and intelligent.

3.1 FLEXIBILITY

Flexibility refers to the ability to change or be changed easily in response to the situation. When it comes to systems, flexibility is not about freedom of choice from preprogrammed actions, but about autonomously reshaping itself. Another way to put it is the ability to reconfigure in pursuit of given goals and constraints. Given that it is not possible to cover all potential scenarios during the design phase, systems must be able to self-adapt to current situations and environments.
Flexible systems reduce risks and maximize opportunities. Their ability to adapt enables them to resume operation when failures occur or if resources are unavailable, provided that alternatives exist. At the same time, they embrace new capabilities and maximize performance under favorable circumstances [14]. A flexible system uses its available resources to provide the best possible service at any time.

3.2 INTELLIGENCE

Intelligence refers to the ability to acquire and apply knowledge and skills. At the Fraunhofer Institute for Cognitive Systems IKS, we view cognitive systems as technical systems capable of independently solving and developing strategies for human tasks. To accomplish this, these systems are equipped with cognitive capabilities for context comprehension, interaction, adaptation and learning. Cognitive systems can utilize artificial intelligence (AI) methods such as machine learning, neural networks and deep learning, but rely on other approaches as well.Cognitive systems are characterized by continuous adaptation to the current context with the goal of improving. While it is nearly impossible to design an optimal system that achieves the best performance in all situations, such systems can operate in a constantly changing environment and improve on their own. They can even learn to manage situations that were not taken into consideration during the design phase.
A cognitive system uses its available resources to learn from experience and modify its algorithms to improve future decisions.

3.3 RESILIENCE

Resilience is the persistence of dependability when facing changes [15]. It refers to the ability to manage changing environmental conditions, which in most cases are not favorable, and to ensure safety and solid performance at all times. Resilient systems must also be able to gracefully handle situations when various failures occur and resources are unavailable. This extends beyond simple fault-tolerance to include handling unexpected situations that were not considered during the design.
Dependability promises uninterrupted service that conforms to the desired level of quality. By definition, resilience means that predicting and implementing appropriate measures and responses for all possible situations in advance is unrealistic. The system must therefore continuously adapt to the current context to maintain uninterrupted operation at the required safety level. To do this, it has to be aware of how its resources support different contexts and continuously refine this knowledge. A resilient system uses its available resources to provide the most robust and dependable services possible.

3.4 FLEXILIENCE

We introduce the term **flexilience** to refer to the quality of a system that comprises the aspects previously outlined in this chapter: flexibility, resilience, and intelligence. However, each aspect has to strive to utilize resources differently, which makes it difficult to optimize them at the same time.
A flexible system uses its available resources to provide the best possible service at any time. A cognitive system uses its available resources to learn from experience and modify its algorithms to improve future decision-making.
A resilient system uses its available resources to provide the most robust and dependable service possible.
A flexilient system has to find a unique compromise to address its current challenges.
The triangle of key qualities in *Figure 2* helps to identify the direction in which the system needs to be tuned. For example, performance can be improved by introducing more flexibility and intelligence, which enables the system to adapt to the current context, learn from experience and thus provide the best performance, regardless if the current scenario was taken into account during the design or if all resources are available.
If high safety levels are necessary and the system has critical, life-dependant functions, it would switch to a safe stop state, always providing safety, but very low reliability or performance. The system might be operating only in a narrow range of scenarios defined during design time, which could be extended to increase the level of resilience and intelligence.

While flexibility can also improve reliability and performance, it must be employed with discretion so as not to interfere with knowledge regarding potential faults and to what extent the system is dependable enough to tolerate them, unless this information is also updated. The downside is that this increases the cost of developing and validating the system.

Flexilience refers to this area of conflict and the challenge in finding the right balance. Our approach proposes how the requirements and configurations of such systems can be analyzed and validated. Moreover, by making this information available to the system at runtime, it can become self-aware and be in a position to handle unexpected situations. This approach is detailed in the next chapter.

Figure 2: Key qualities of dependable cloud-based cyber-physical systems.

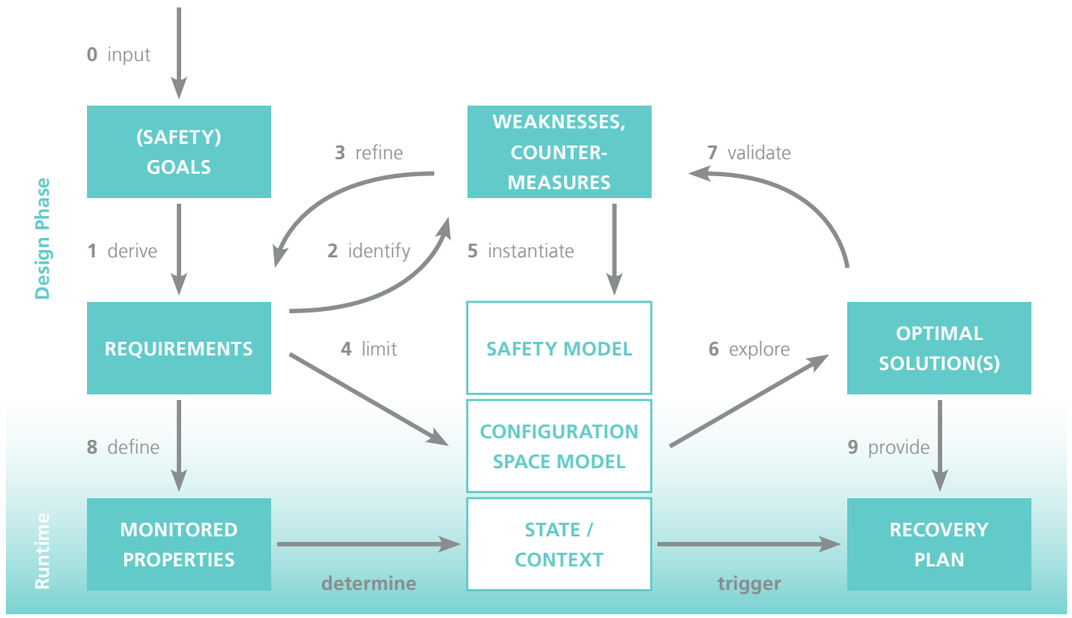
4 OUR SOLUTIONS FOR FLEXILIENT END-TO-END ARCHITECTURES

This paper has so far outlined the challenges involved with dependable cloud-based cyber-physical systems and presented flexilience as a desired quality. This chapter introduces our solutions for creating flexilient end-to-end architectures and describes how we can help our customers to develop them. While we utilize one specific example to illustrate the overall process, the described methods can be integrated into any existing process.

4.1 PROCESS OVERVIEW

An overview of the process for analyzing and validating flexilient end-to-end architectures is illustrated in *Figure 3*. The various artifacts shown in the process, such as system requirements or optimal configurations, can serve as the input and output of the various actions. The process includes various feedback cycles that allow for iterative and continuous improvement and refinement of the designed architecture. The service that we offer involves applying the methods (mainly for case studies or prototypes) and providing the methods that facilitate these actions to our customers.

Figure 3: Process carried out to design and evaluate the architecture approach



The system goals, especially those related but not limited to safety, serve as the initial *input (0)*. While these goals are normally defined by our customers, Fraunhofer IKS can provide assistance in creating them through methods such as hazard and risk analyses (HARA). The initial input influences all the subsequent architecture design decisions with the aim of satisfying or maximizing the goals.

In the next step, **requirements** for the system are *derived (1)* from the goals. Starting from top-level requirements and constraints for the overall system, the basic components and associated task descriptions are then derived and analyzed as requirements in text form. This analysis systematically *identifies (2)* the failure modes and, more generally, **weaknesses**. It also *refines (3)* the requirements by suggesting **counter measures** that enable fault prevention and fault tolerance or otherwise address relevant weaknesses. Our analysis approach is discussed in more detail when we describe our weakness-driven requirements refinement in Section 4.2.

The requirements *limit (4)* the possible configuration space of the end-to-end architecture. What we offer is to implement a domain-specific system model that includes the corresponding context as a way to describe the specific scenarios under analysis. This is the **configuration space model** – a set of tools to describe the degrees of freedom and their constraints, collect information, perform analyses and produce a formalized description of the goals and requirements. An integral, yet distinct part is a **safety model**, which makes it possible to perform safety-oriented analyses and set limits to the performance-related optimizations. The models can be *instantiated (5)* and narrowed down in order to further analyze selected weaknesses, among other things. **Solutions** to these models are *explored (6)* in order to identify and evaluate (optimal) system configuration candidates.

While the optimal solutions will be constructed to fulfill the specified requirements, it must be *validated (7)* to make sure that the defined goals are in fact satisfied and that no weaknesses have been overlooked during requirements refinement. Our service offer entails carrying out detailed analyses and simulations to evaluate end-to-end architecture and application solutions, which involves validating the safety and performance of the solutions with respect to the defined goals and also providing fault forecasts or help in identifying critical scenarios. We can also offer to help bring the design concept to runtime using a monitor and recovery concept. The **monitored properties** are *defined (8)* by the requirements and the **recovery plans** are *provided (9)* by optimal solutions for the identified contexts. At runtime, the monitored properties are used to *determine* the current **state and context** that *trigger* a recovery plan if needed. Our offer does not stop at these artifacts, but can also include concepts and prototypes for the required monitors and recovery mechanisms that constitute the system's self-awareness.

Currently, knowledge is mainly introduced by manual engineering steps in the design phase. One focus of our research is automation of these steps, eventually making it possible to move some steps to the runtime phase and as a result further increasing the degree of autonomy in flexilient systems.

4.2 WEAKNESS-DRIVEN REQUIREMENTS REFINEMENT

The weakness-driven requirements refinement is an iterative process for uncovering the system's weaknesses and integrating countermeasures along the refinement. A *weakness* is *any deviation from the system's intended function*, such as a potential safety hazard or failure, a security threat, or a breach of performance thresholds. The general intent is to identify potential weaknesses in the system-of-systems and determine what conditions are necessary to handle them on subsystem level. The input to this phase is the top-level requirements and a draft of the system.

The main purpose of this input is to guide the refinement and decomposition. As such, they can be defined informally. Along with the requirements for each subsystem, potential weaknesses are identified using a HAZOP-oriented process [16], such as by applying a set of guide-words to the requirement descriptions. Examples for guide words include *not, more, less, as well as, part of, reverse, other than, early, late, before* and *after*. To scrutinize the consistency of the requirements, the following questions are used to identify weaknesses in addition to common HAZOP guide-words:

— Internal: How can the subject itself fail to fulfill the requirement?

— External: What external influences can cause the subject to fail (the intent of) the requirement?

— Integrity: Are there any terms, definitions or values used by the requirement that can impact the intent if chosen incorrectly?

Each of the identified weaknesses has to be resolved by verification, assumptions, or other requirements. A *verification* describes a method to verify why a weakness will not occur or is mitigated sufficiently. An *assumption* is a statement describing a property that is assumed to be valid. Therefore, assumptions are formulated to document parts of the system that are expected to work or resolved by the individual subsystem. For example, we assume a correct implementation of requirements but will not specify how this is achieved.

Other requirements that resolve weaknesses either refine affected requirements to detail how the failure is avoided, or impose additional requirements to mitigate the cause or effect of certain faults. New requirements may introduce new weaknesses that are identified and mitigated in further iterations. This is continued until all weaknesses are resolved. The resulting requirements describe the roles of subsystems in the system-of-systems and their interfaces. By recording bidirectional relations between requirements, weaknesses and resolutions, validation of an implementation is facilitated if the reason for a requirement can be easily traced.

We therefore implemented a domain-specific language to support the approach. Our service involves carrying out and adapting this process for our customers or helping them to exploit it.

4.3 PLASTIC ARCHITECTURES

Depending on available resources and connected end-users, the system's architecture may need to change dynamically. For example, depending on the geographic location of the mobile system, potential faults in the connection, or other issues can mean that cloud- or edge-services or other resources are temporarily unavailable. However, system operability, performance and safety have to be maintained at satisfactory levels. The resources should be utilized whenever they become available again. In this sense, the architecture is plastic because its structure and boundaries change over time. Plastic architectures are an essential part of systems to ensure flexibility and interoperability.

Architectures for resilient cognitive systems-of-systems can potentially have three levels: global (cloud services), regional (edge services) and local (embedded systems). Challenges arise when changing the system boundaries across the layers. Embedded on-board systems provide real-time performance. Operational decisions are made within the timeframe of milliseconds and basic functionalities are provided, such as monitoring and actuator controls. Edge services provide *adaptive dependability management*, which we describe in Section 4.6.2. Tactical decisions are made within a matter of seconds and minutes in accordance with the availability of the local servers. Cloud services provide continuous safety management. Strategic decisions, which are made over the course of days and months, are best rendered in line with the performance of the server clusters.

Utilizing our experience [17], we help customers integrate these types of constantly-changing aspects into the requirements by considering different levels of abstractions that allow generalizing certain degrees of freedom and defining the timeframe in which reactions need to be performed. We can also provide patterns that for facilitating implementation and creating a safety case.

4.4 DEGRADATION AND UPGRADES

One of the crucial features of resilient systems is maintaining operability and a minimum level of service at all times and without interruptions. To achieve this dependability, the system must be able to continue operation if resources or services become unavailable. With this in mind, we rely on a degradation and upgrade concept that refers to the ability of a system to gracefully degrade its own functionality in such a manner that available resources and services are sufficient [18]. As soon as resources and services are restored, the system can perform an upgrade and return to full performance. While this type of behavior needs to be managed dynamically based on the current situation, it can still be taken into account and planned for during the design phase if the involved capabilities are known [19]. While preplanning degradation limits the potential degree of flexibility, it facilitates the integration of recovery in line with existing safety standards. Furthermore, degradation planning always needs to be performed before it is needed, otherwise the system may run out of resources before it is able to adapt.

The concept can be illustrated with a simple example. Consider the topology of an architecture with a mobile cyber-physical system, such as an autonomous vehicle or a driverless industrial truck, including cloud services, edge services and communication with other users as depicted in *Figure 4*. When all services and resources are available, including the cloud services, the system provides full functionality within a global context. In case of a communication failure, the mobile system needs to gracefully degrade to its ego context, where only information from its own sensors is available. The furthest degradation possible is to a minimum set of functions that ensure safety, but potentially provide only limited functionality, such as performing a minimum risk maneuver. This is why the more basic services that are required for a fallback strategy have to be available in the mobile system (cf. *Figure 5*). Through our experience with flexilient systems, we can support our customers with design patterns that facilitate shaping different performance levels based on available resources and ensuring safety.

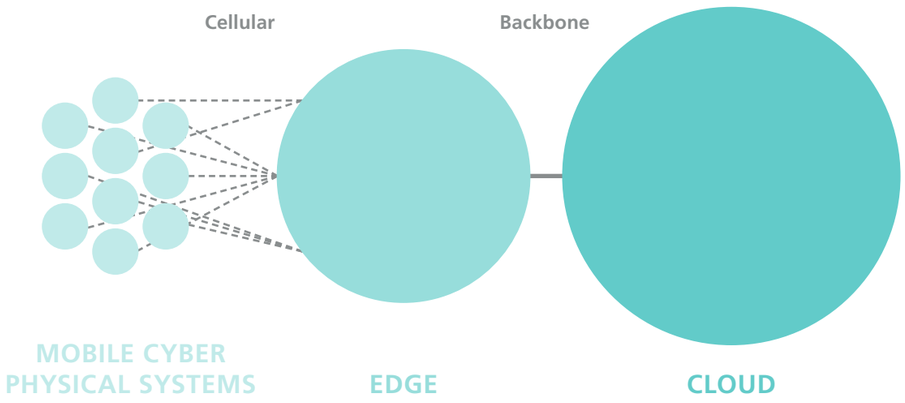


Figure 4: Generic topology of the system

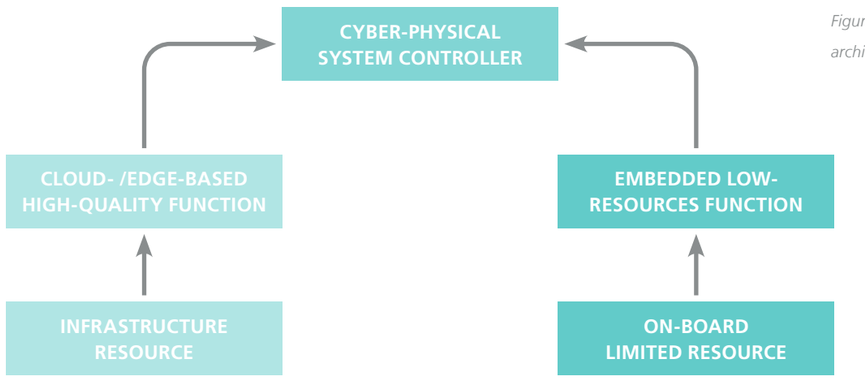


Figure 5: Example of application architecture of the system

4.5 UPDATES

In today's world, users expect software — especially when offered as a service — to be continuously updated in order to improve reliability, safety or to add new functions. Depending on the scale of such changes, the design process can be more similar to a completely new solution than a simple iteration.
In any case, such updates must be included in the life-cycle of modern cloud-based systems.
Flexilience can provide the means to transactionally switch from old to new code, such as by preparing a recovery plan that will switch to an updated service when the system is in a state that permits a safe update.
Since the involved services can be spread across different parts of the end-to-end architecture, and given that not all subsystems are able to update at once, changes must be coordinated across the different systems to ensure a safe and sound configuration. We can help our customers avoid these and other pitfalls when updating their own software.
Moreover, existing safety standards may not include the means to consider update mechanisms. We can help our customers build a safety case that can be used as a justification for approving such mechanisms.

4.6 SELF-AWARENESS

Self-awareness is necessary when designing autonomous systems. These types of systems must be able to perform self-assessment to ensure the proper detection and handling of failures so that the required performance and safety is maintained regardless of the situation. Self-awareness can be defined as the ability of the system to determine its own state, detect faults and identify possible actions and the corresponding results within the system itself and within its environment.
The first kinds of systems that will be able to offer true self-awareness are those that have the necessary amount of processing power available in cloud solutions. Nevertheless, these systems also contain embedded devices with limited capabilities that must operate safely even if the connection to the cloud is not available. These systems must have some degree of basic self-awareness as well. With this in mind, we recommend a predefined monitor and recovery approach that emulates self-awareness for the given context, in addition to more elaborate adaptive dependability management [20].

4.6.1 MONITORING AND RECOVERY

The most basic type of dynamic situation management involves defining triggers that cause specific actions. Monitors are utilized to detect when the system's current configuration needs to change and start the identified recovery mechanisms. A common way to articulate the context an automated system is designed to operate in, is to describe the operational design domain (ODD). An ODD defines the domain (including all external and internal conditions such as type of road, weather, speed, as well as status of the sensors and actuators in the case of an autonomous vehicle) over which the system can operate safely. Triggers define thresholds along the borders of this domain, and designate when the system needs to adapt and potentially degrade more advanced functionality to ensure safety. In addition, a restricted operational domain (ROD) describes the domain in which the system can currently operate safely, such as an ODD that is degraded due to sensor failure.

This concept, however appealing and promising, can become complex and presents numerous challenges when considering not only a single vehicle or a system, but several connected vehicles communicating with each other and with an infrastructure. Similar challenges arise for mobile robots and infrastructures such as traffic management systems that operate at multiple locations in coordination. Apart from the opportunities this concept yields, it also brings with it various issues, such as how to define a common operational design domain for many participants or how to manage a shared model of reality. As described for the overall design process, the functionality must be addressed holistically in order to take into account the interaction of the various influences across multiple components. Afterwards, weaknesses can be broken down and necessary actions identified for each individual sub-system. Even when facing unexpected deviations, subsystem monitors must keep track of the current overall system state [21]. As part of our services, we can help customers identify these monitoring and recovery approaches and provide insights into which cases such an approach is feasible or when dynamic management is advisable.

4.6.2 ADAPTIVE DEPENDABILITY MANAGEMENT

Resilience — the capability to guarantee system dependability at all times regardless of the situation in dynamically changing environments is an essential feature of the systems described in this paper. Designing a system that is prepared for any conceivable scenario is a highly difficult, if not impossible task that would eventually result in a very limited system featuring only a few basic functions that offer low performance or which switch to fail-stop mode when an unexpected situation is encountered. While a simple monitoring and recovery approach can provide good performance if the majority of contexts and reactions can be predicted, more sophisticated methods become necessary as the level of complexity rises. A truly flexilient system reacts and adapts to the current situation by taking into account the available resources, active and potential faults, as well as the overall context. Such systems can achieve a high level of performance under favorable conditions, and operate safely e ven when situations turn disadvantageous.
Self-adaptation cycles consist of constant monitoring, updating the system's model of its own state and safety, as well as continuous model-based analysis with system optimization and adaptation. Typical loop-based control patterns are MAPE (monitoring, analysis, planning and execution) and SPA (sense, plan and act), which are typically combined with knowledge that we like to represent in various kinds of models.
Fundamental for adaptive dependability management is the self-awareness of the system in order to correctly evaluate the current situation and the effects of reactions. To ensure safety and enable adaptive safety and dependability management, a safety model is designed, maintained and updated at runtime. This includes calculating current capability space (possible system configurations derived from the current configuration and data provided by self-monitoring of the system), a context specific goal space (system configurations and restrictions required to meet the objectives derived from current situation in an external context), potential hazards and the current safety goal.

The safety model enables adaptive safety management, for instance by selecting an optimized system configuration from the intersection of the current goal space and current capability space [22]. As there is still a lot of research being carried out in this area, Fraunhofer IKS can conduct targeted re-search in a search for solutions to address specific scenarios using state-of-the-art techniques or adapt them to a specific use case.

5 CASE STUDIES

Flexilience – a novel approach to find a unique balance between resilience, intelligence, and flexibility allows numerous challenges to be overcome that cloud-based dependable systems-of-systems must face. We propose various techniques that enable the development of flexilient systems, such as weakness-driven requi-rements refinement, plastic architectures, graceful degradations and upgrades, safe updates, and increasing self-awareness of the system. The process that we have outlined here for analyzing and validating flexilient end-to-end architectures permits companies to take an iterative and continuous approach to development.
The process details and selection of the techniques used to achieve flexilient design can be adapted to a specific application scenario. While cloud-based approaches often promise manifold advan-tages over local solutions, additional information is usually required prior to making a decision on which option is the best.
As part of our services, we can apply this process to a case study that explores specific scenarios or contribute findings from our experience with similar cases in the past. For example, simulating the main aspects of a function permits a comparison of the efficiency of different approaches, which can help in evaluating and quantifying the potential benefit of a cloud-based function while uncovering the potential risks and weaknesses. The required measures and associated implementation costs can then be estimated using the case study as a basis.

6 ACKNOWLEDGEMENT

This work was funded by the Bavarian Ministry for Economic Affairs, Regional Development and Energy as part of a project to support the thematic development of the Institute for Cognitive Systems.

Sponsored by:

Bavarian Ministry of Economic Affairs, Regional Development and Energy

IMPRINT

REFERENCES

[1] E. Uhlemann, "Time for autonomous vehicles to connect [connected vehicles]," *IEEE Vehicular Technology Magazine*, vol. 13, no. 3, pp. 10–13, 2018.

[2] ISO/IEC/IEEE 42010:2011(en), "Systems and software engineering — Architecture description," 2011.

[3] V. Schönemann, M. Duschek, and H. Winner, "Maneuver-based Adaptive Safety Zone for Infrastructure-Supported Automated Valet Parking," in *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems*, pp. 343–351, 2019.

[4] A. Gharaibeh, M. A. Salahuddin, S. J. Hussini, A. Khreishah, I. Khalil, M. Guizani, and A. Al-Fuqaha, "Smart Cities: A Survey on Data Management, Security, and Enabling Technologies," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2456–2501, 2017.

[5] X. Liu, J. Cao, Y. Yang, and S. Jiang, "CPS-Based Smart Warehouse for Industry 4.0: A Survey of the Underlying Technologies," *Computers*, vol. 7, p. 13, Mar. 2018.

[6] ISO 3691-4:2020(E), "Industrial trucks — Safety requirements and verification — Part 4: Driverless industrial trucks and their systems," 2020.

[7] ISO 26262:2018, "Road vehicles — Functional safety," 2018.

[8] ISO/PAS 21448, "Road vehicles — Safety of the intended functionality," 2019.

[9] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[10] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "How to place your apps in the fog: State of the art and open challenges," *Software: Practice and Experience*, Nov. 2019.

[11] B. Di Martino, M. Rak, M. Ficco, A. Esposito, S. Maisto, and S. Nacchia, "Internet of things reference architectures, security and interoperability: A survey," *Internet of Things*, vol. 1-2, pp. 99–112, Sept. 2018.

[12] V. L. L. Thing and J. Wu, "Autonomous vehicle security: A taxonomy of attacks and defences," in *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 164–170, 2016.

[13] H. Seifzadeh, H. Abolhassani, and M. S. Moshkenani, "A survey of dynamic software updating," *Journal of Software: Evolution and Process*, vol. 25, no. 5, pp. 535–568, 2013.

[14] R. de Neufville and S. Scholtes, *Flexibility in engineering design*. MIT Press, 2011.

[15] J.-C. Laprie, "From Dependability to Resilience," DSN, 2008.

[16] IEC 61882:2016, "Hazard and Operability studies (HAZOP studies) - Application guide," 2016.

[17] Fraunhofer IKS, "Projects and References." https://www.iks.fraunhofer.de/en/projects.html

[18] C. Drabek and D. Seydel, "Sicherheitskritische Software-Funktionen sind in der Edge oder Cloud gut aufgehoben," 2020. https://safe-intelligence.fraunhofer.de/absicherung-von-verteilungskonzepten

[19] P. Schleiss, C. Drabek, G. Weiss, and B. Bauer, "Generic management of availability in fail-operational automotive systems," in *International Conference on Computer Safety, Reliability, and Security*, pp. 179–194, 2017.

[20] M. Trapp, D. Schneider, and G. Weiss, "Towards safety-awareness and dynamic safety management," in *14th European Dependable Computing Conference (EDCC)*, pp. 107–111, 2018.

[21] C. Drabek, G. Weiss, and B. Bauer, "Resumption of runtime verification monitors: Method, approach and application," *International Journal on Advances in Software*, vol. 11, wno. 1 and 2, pp. 18–33, 2018.

[22] M. Trapp and G. Weiß, "Towards dynamic safety management for autonomous systems," in Engineering Safe Autonomy: *Proceedings of the 27th Safety-Critical Systems Symposium (SCSC)*, pp. 193–204, 2019.

AUTHORS

Fraunhofer IKS
Christian Drabek
christian.drabek@iks.fraunhofer.de

Anna Kosmalska
anna.kosmalska@iks.fraunhofer.de

EDITOR

Fraunhofer Institute for Cognitive Systems IKS
Hansastr. 32
D-80686 Munich

Phone: +49 089 547088-0
Fax: +49 089 547088-220
info@iks.fraunhofer.de
www.iks.fraunhofer.de

Miriam Friedmann
Phone: +49 89 547088-351
miriam.friedmann@iks.fraunhofer.de

IMAGE CREDITS

Cover image & key visual: © istock / wingmar
Page 3: © istock / kokouu
Page 3: © istock / Roberto Palmer
Page 4: © istock / Thossaphol
Page 6: © istock / AlxeyPnferov
Page 4-6: © istock / chuyu
Page 7: © istock / EduardHarkonen
Page 8-11: © istock / Bim
Page 12-13: © istock / Lukas Bischoff

READ OUR BLOG!

For news and further topics of Fraunhofer IKS visit our website and our blog:

www.iks.fraunhofer.de | safe-intelligence.fraunhofer.de