Towards Resilience by Self-Adaptation of Industrial Control Systems

Laurin Prenzel
Technical University of Munich
Munich, Germany
laurin.prenzel@tum.de

Sebastian Steinhorst
Technical University of Munich
Munich, Germany
sebastian.steinhorst@tum.de

***Abstract*—Resilience is a critical quality of future Industrial Control Systems (ICS). The ability to detect and react to unanticipated attacks, bugs, and failures is crucial. Self-adaptation can provide this ability, yet it is difficult to achieve in safety-critical real-time systems, since strict safety and timing requirements must be guaranteed. Recent results indicate that automated adaptation of ICS using the IEC 61499 is possible, however it has not been analyzed how much dynamic adaptation can contribute to overall system resilience. In this paper, we analyze how dynamic adaptation can be embedded into industrial control architectures, and quantify its advantage over a traditional restart. We propose a self-adaptive architecture using the MAPE-K model and merge it with the existing models for ICS. Using measurements on a real system, we estimate the expected adaptation time of selected adaptation scenarios and calculate the loss of productivity depending on the reaction time and adaptation complexity. The results show that using current dynamic adaptation mechanisms, minor to moderate adaptations can be completed within 10 ms, while larger adaptations can take up to a second from initialisation to cleanup. The resilience gain is larger the faster the reaction is initiated, which indicates that once dynamic adaptation is available, a faster detection and decision-making becomes more important. Dynamic adaptation can provide ICS the means to evolve and react rapidly, preparing them for an agile, flexible, and resilient future.**

***Index Terms*—Dynamic Reconfiguration, Downtimeless System Evolution, Resilient Industrial Control System**

I. Introduction

Resilience is a trait that sets the human apart from the machine. It describes the fundamental ability of surviving and overcoming hardship. In technical or industrial systems, we strive to embed this ability into the machine to make it robust against unexpected failures, bugs, or attacks. While current industrial control architectures are designed to be reliable (e.g. against expected failures), resilience is often overlooked. In other domains, autonomic computation and self-adaptation, which permit resilience, are openly being discussed [1, 2]. The Industrial Internet of Things, for instance, introduces an abundance of resilience potentials (in the form of interconnections and redundancies) [3]. Current static industrial control architectures struggle to incorporate these potentials due to their inflexibility and rigidity, thus preventing resilience.

Reconfigurable and flexible manufacturing systems have been proposed to handle the frequent changes in requirements, and to make manufacturing systems more cost effective [4]. These high-level concepts are mostly concerned with quickly adjusting production capacity and functionalities according to market demand. The use of reconfiguration and flexibility to improve safety and availability of functionality, on the other hand, is not commonly addressed. By contrast, autonomic computation and self-adapting architectures are closely related to fault-tolerance mechanisms. The MAPE feedback loop (Monitor, Analyze, Plan, Execute) is designed to handle unanticipated changes, faults, or attacks [1, 5].

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany (BMBF) in the framework of ReMiX (project number 01IS18063B).

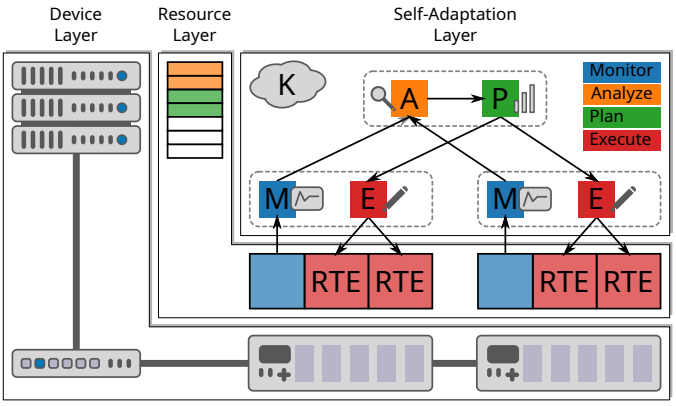


Fig. 1. The IEC 61499 provides device and resource layer models to separate the control application from the resources. The MAPE-K model can add a self-adaptation layer that provides resilience, flexibility, and agility.

The key to self-adaptation is the ability to adapt, which current ICS struggle to achieve. Industrial standards, such as the IEC 61499, enable dynamic reconfiguration or adaptation of distributed industrial control software, yet it is rarely used since the standard does not explain how to achieve it safely and in real-time. Recent results in research, however, indicate how real-time dynamic adaptation can be implemented safely [6, 7]. Yet, there is currently no consensus on when and how dynamic adaptation can or should be used in practice.

In this paper, we propose a self-adaptive architecture for ICS, as displayed in Figure 1, which merges the MAPE-K feedback loop and the IEC 61499 models for ICS. Dynamic adaptation on the runtime level enables distributed self-adaptation. This architecture promises to provide superior resilience and flexibility, and this paper aims to assess the resilience gain introduced by dynamic adaptation. Thus, we address the following research question: *What is the quantitative impact of dynamic adaptation on the system resilience of Industrial Control Systems?*

We briefly introduce the relevant origins of (self-) adaptation and outline the state of the art in dynamic adaptation of ICS in Section II. Section III establishes the concepts of resilience,

and how this metric can be quantified. We propose the resilient ICS architecture in Section IV, and highlight the phases of dynamic adaptation. In Section V, we evaluate the potential of dynamic adaptation by providing measurements, extrapolating the required adaptation times, and quantifying the resilience gain of dynamic adaptation. Further research directions are proposed in Section VI, and we conclude in Section VII.

II. Background

In this section, we introduce the concepts of dependability and fault-tolerance, before advancing to self-adaptation. We then shortly summarize the state of the art in dynamic adaptation for ICS that we build upon.

A. Dependability & Fault-Tolerance

A large body of research is devoted to increasing the dependability of technical systems. The term covers all facets of the desire to provide trusted and correct services. It integrates attributes such as *availability*, *reliability*, *safety*, and *integrity* [8]. We use the common definitions of *fault*, *error*, and *failure*: A fault is a (internal or external) flaw, bug, or vulnerability that may eventually lead to an error, i.e. an unintended outcome. An error may lead to a failure, i.e. the inability to perform the required service [9]. Thus, the goal of dependable system design is to reduce the number of faults, errors, and failures.

Generally, dependability can be increased in numerous ways. Apart from prevention, forecasting, and removal of faults, fault-tolerance is the concept of preserving a service in the presence of faults [8]. It is often used together with other terms such as *fail-safe* or *fail-operational* behavior, and their definitions can vary [10]. We do not differentiate between these terms.

In its essence, fault-tolerance is the exploitation of redundancy, i.e. having more resources (among some dimension) than necessary [9]. This allows for the compensation of a fault, e.g. by reallocating resources, switching to a replacement, or degrading non-essential functionalities. Many of these mechanisms are implemented beforehand, and resources are allocated in anticipation of a fault. Yet, in technical applications, it is infeasible to anticipate and prevent all faults. Over time, components degrade, requirements change, and the environment transforms. Adaptation in response to these changes is a desirable property in technical systems and may lead to resilient behavior.

B. Self-Adaptation

The bottom-line of fault-tolerance is to achieve a satisfactory level of autonomy when facing a particular fault. This vision has been established in research, and while progress has been made, it has not been achieved [11]. A common approach to self-adaptation is the MAPE-K model (Monitoring, Analysis, Planning, Execution over Knowledge) [1, 5]. An integration of this model into an industrial automation system is displayed in Figure 1. The knowledge (K) represents all data and information shared between components, such as the particular configurations, or adaptation goals. During the monitoring (M), further information is gathered that contributes to the knowledge. During analysis (A), the knowledge is examined to determine whether or not an adaptation is necessary. If so, the planning (P) phase allows the preparation and development of the adaptation procedure, which will be initiated and completed during the execution (E) phase.

This model is suitable for both centralized, as well as decentralized architectures [12]. The architecture displayed in Figure 1 is a hybrid model, in which the monitoring and execution is decentralized, whereas the (computationally heavy) analysis and planning phases are performed on a central infrastructure. This is similar to the approach in [13], where the control devices are active participants in the decision-making process through a consensus. In this paper, we focus on the adaptation capability, thus centralized decision-making is sufficient.

The integral element of any self-adaptation model is the ability to adapt, which is particularly difficult for stateful systems with real-time constraints, such as ICS. In the next section, we illustrate the state of the art on dynamic adaptation of ICS, in contrast to non-dynamic adaptation, i.e. a restart of some form.

C. Dynamic Adaptation of Industrial Control Systems

Industrial Control Systems (ICS) are safety-critical real-time systems. As such, they must satisfy strict requirements and must undergo extensive verification and validation. In [14], the potentials of self-adaptation in ICS are detailed, specifically for typical IEC 61131-3 POUs / tasks. A key insight is that the adaptation may disturb the execution, thus the advantage of an adaptation must be weighed against the risks of a delay. This is particularly important for traditional ICS with computationally heavy, monolithic tasks.

At this point, it is convenient to introduce the IEC 61499, which extends the models of the IEC 61131-3, with the benefit of splitting the monolithic state to facilitate distribution and reconfiguration [15]. All state is encapsulated inside function blocks (FBs), and execution is event-triggered instead of cyclic. Dynamic reconfiguration of IEC 61499 applications has been proposed since its inception, yet it's rarely applied in practice. Two main challenges of dynamic reconfiguration are the correctness or preservation of consistency, and the satisfaction of real-time constraints [15]. We do not distinguish between dynamic *adaptation* and dynamic *reconfiguration* and use the terms interchangably.

Recently, it has been shown that it is possible to automatically generate reconfiguration sequences for a variety of reconfiguration scenarios, while preserving the consistency of the execution behaviors [6]. This is facilitated by the ability to precisely modify only the affected FBs, and the event-triggered execution. While for many scenarios no further information is required, some adaptations require details, e.g. regarding the necessary state transformation. For self-adaptation, this information could be present in the knowledge base.

The real-time execution of the IEC 61499 was addressed by [16]. More recently, it was shown in [7] that the schedulability of an ICS using IEC 61499 under reconfiguration can be decided for Rate Monotonic (RM) scheduling using the schedulability condition

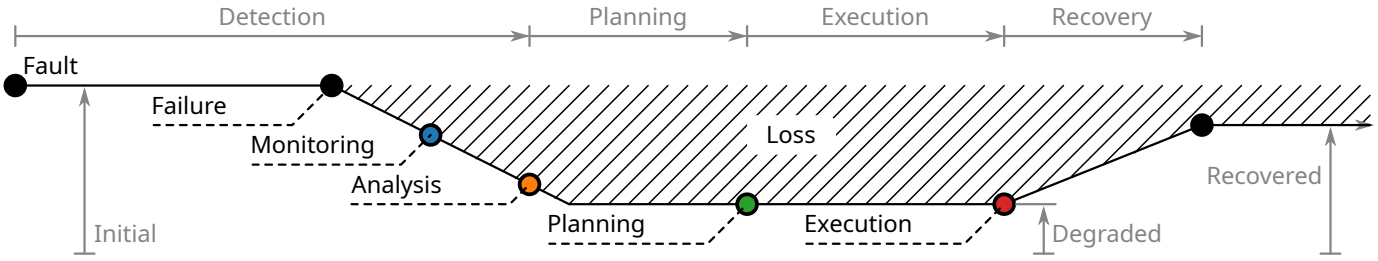


Fig. 2. The resilience graph allows the quantification of a resilience loss over time. The MAPE-K model can be integrated into this graph to highlight how the phases of the model affect the resilience loss.

$$\forall i, 1 \leq i \leq n, \quad \underbrace{\frac{C_i}{T_i}}_{\text{Execution}} + \underbrace{\frac{B_i}{T_i}}_{\text{Blocking}} + \underbrace{\sum_{j=0}^{i-1} \frac{C_j}{T_j}}_{\text{Preemption}} \leq \underbrace{i(2^{1/i} - 1)}_{\text{Max. Utilization}}, \quad (1)$$

where C_i is the execution time, B_i is the blocking time, and T_i is the rate and deadline for n real-time tasks sorted by their rate. Priority inversion for shared resources and suspension must be avoided using a Priority Ceiling Protocol (PCP). The blocking time B_i is defined by the blocking by shared access to FBs, and the suspension caused by a reconfiguration sequence.

As a consequence, dynamic adaptation of ICS without violating the real-time constraints is currently possible. In this paper, we investigate the feasibility of dynamic adaptation and quantify the potential benefits over an offline adaptation, i.e. a restart.

III. Resilience Quantification

Measures of fault-tolerance commonly rely on qualitative differences, i.e. the system is fault-tolerant of a specific fault or not. Thus, they struggle to quantify degradation [9]. By contrast, *resilience* is commonly used in a quantitative context, i.e. a person / system is more resilient than another. In this paper, we use an established resilience metric to quantitatively assess the impact of dynamic adaptation in response to a fault.

Various definitions of resilience from numerous domains are outlined in [17]. We use the following definition: *"Resilience is the ability to prevent something bad from happening, Or the ability to prevent something bad from becoming worse, Or the ability to recover from something bad once it has happened."* [18]. There are three components in this definition: Prevention, survival, and recovery. These distinct behaviors will emerge again in the evaluation as part of different scenarios.

Resilience can be quantified as a function of time and visualized in a resilience graph [19]. An example is displayed in Figure 2, where the phases of the MAPE loop are inserted. [19] identified three distinct system states: The *original* state, a *disrupted* or *degraded* state, and a *recovered* state. In our specific example, the *original* state ends with the failure, and the *degraded* state ends when an adaptation is executed. In practical applications, the graph may be arbitrarily complex.

A failure generally leads to a loss in some metric. If a failure does not create any loss, then no action is required. Computation of a loss requires the definition of a *figure of merit* $F(\bullet)$ [19]. For the domain of industrial control, possible metrics could be *productivity*, *process quality*, or *process / communication delays*. Since we focus on the methodological aspects of resilience quantification, for the rest of the paper, we assume a *figure of merit* $F(\bullet)$ for which increasing values are preferred, and we assume that the system can be sufficiently quantified using a single metric. For practical applications, multiple metrics may be necessary to fully capture the complexities.

A. Calculation

Using the aforementioned metric, it is possible to calculate the loss of this metric over time. [20] introduces the calculation of a *resilience loss*

$$RL = \int_{t_0}^{t_1} [100 - Q(t)] \mathrm{d}t \quad (2)$$

by integrating the loss of quality over time, where the quality $Q(t)$ is given as a value from 0 to 100. A smaller resilience loss would thus indicate higher resilience. For our purposes, we choose an abstract quality of service, e.g. productivity, as our *figure of merit*, which can be measured in percent. By integrating this metric over time, we receive the lost productivity in the unit of time. For example, if a failure causes a total loss of function for 5 seconds, this would lead to an equivalent resilience loss as a partial loss of 50% for 10 seconds.

A difficulty of resilience quantification is the choice of metric, and the comparison between metrics. Mitigating a loss in one metric by compensating it with another metric requires a conversion factor that is often hard to derive. In our case, we can assume that a fault will lead to a failure of the ICS, which will directly cause a loss of productivity. We explicitly exclude faults and failures that must immediately lead to a full shutdown to prevent catastrophic consequences, for example an emergency stop, since there can be no meaningful quantification of losses in this case. This does not mean that dynamic adaptation can not be used to deal with these faults and failures, if anything it may be the only reasonable solution for true non-stop systems. However, the added risk of failure during the adaptation must be carefully considered.

IV. Role of Adaptation in Resilient Industrial Control Systems

In this section, we demonstrate the signficance of self-adaptation for resilient ICS, and introduce a self-adaptive architecture for industrial control based on the existing concepts of the IEC 61499. In the next section, we will quantify the adaptation durations and system resilience.

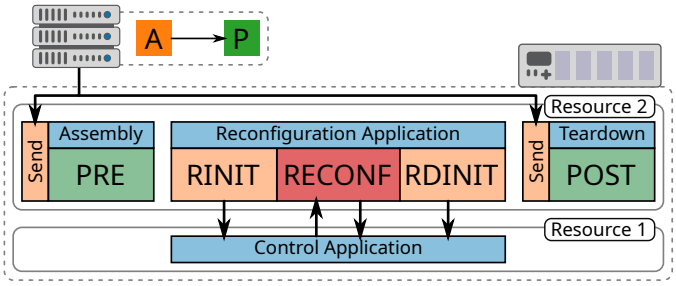


Fig. 3. Once a reaction is decided, the RCA must be assembled on a separate resource. During its execution, it interacts with the control application. Eventually, it can be disassembled in the teardown phase.

A. Self-Adaptation Lifecycle

A common self-adaptation model is the MAPE (Monitor, Analyze, Plan, Execute) loop [1, 5]. It consists of the four aforementioned phases, and can be implemented in various design patterns [12]. Figure 1 introduced a possible architecture, in which the monitoring and execution phases are distributed, and the anlysis and planning is centralized. This causes minimal overhead on the real-time execution, and allows sophisticated methods to be used for analysis and planning.

In this paper, we mainly focus on the execution, thus we split the MAPE cycle in two separate phases: A detection and decision-making phase (MAP), and the execution phase. We can calculate the duration of the two phases as

$$d_{\text{MAP}} = d_M + d_A + d_P, \tag{3}$$

where d_M, d_A, and d_P are the durations of the first three phases, and

$$d_{\text{MAPE}} = d_{\text{MAP}} + d_E, \tag{4}$$

where d_E is the additional duration of the execution phase. The duration d_{MAP} is thus the timespan between the manifestation of the fault, and the start of the reaction to it. The full duration d_{MAPE} ends when the reaction is fully implemented. In the following, we are mostly concerned with the impact of the adaptation duration d_E, and how it affects the system behavior and resilience.

B. Adaptation Phases

The adaptation of ICS using the IEC 61499 can be split into five phases [21] that deal with the assembly, execution, and teardown of a Reconfiguration Control Application (RCA) (Figure 3). The RCA is executed in parallel with the control application, and modifies it on the fly. The five adaptation phases are:

PRE The RCA is transmitted to the control device and assembled in a separate resource using the same operations that are used for the reconfiguration. This phase is non-critical and can be performed over a long period of time.

RINIT After the RCA is started in parallel with the control application, non-critical operations, such as the addition of FBs, can be performed. At this stage, the RCA is executed concurrently with the control application, but with a lower priority.

RECONF In this critical phase, parts of the control application are suspended to prevent unpredictable state changes, and the real-time behavior is disturbed. In a well-designed RCA, the disturbance must not cause the violation of a deadline [7].

RDINIT Once the execution of the control application is resumed, the final non-critical operations remove left-over elements on the application resources, such as removed FBs. This can be performed concurrently with a lower priority.

POST Eventually, the RCA has to be removed to free the resource for further reconfigurations. In this phase, the RCA can either be disassembled, or the entire resource could be deleted.

The duration of the adaptation can be calculated as the sum of the individual durations of each phase:

$$d_E = d_E^{\text{PRE}} + d_E^{\text{RINIT}} + d_E^{\text{RECONF}} + d_E^{\text{RDINIT}} + d_E^{\text{POST}}. \tag{5}$$

Each duration d depends on required execution time C, which is a result of the complexity of the adaptation and is directly affected by the utilization U of the resource. Generally, the exact duration is affected by the scheduling, yet a good estimate can be achieved by dividing the execution time C by the utilization U:

$$d_E^X = \frac{C}{U}. \tag{6}$$

Given that FBs are short-running and must terminate quickly, and RCAs are commonly made up of a large number of FBs, this estimate is accurate for our analysis, since small scheduling differences will average out. In the following section, we calculate the estimated adaptation times for different scenarios using measured execution times.

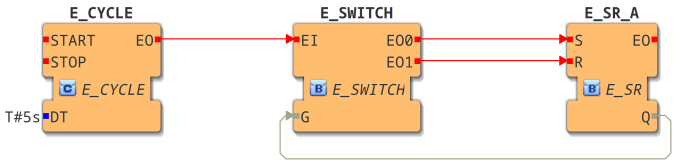
V. Evaluation

We evaluate the impact of dynamic adaptation on the system resilience, and quantify the gain that can be achieved. For this purpose, we compare dynamic adaptation to a traditional restart scenario, in which the system must be first ramped-down, restarted, and finally ramped-up again.

To get a realistic estimate of the required adaptation times, we first measure the execution time of the elements within a simple Reconfiguration Control Application (RCA). Then, we estimate the expected duration of larger adaptations, before we finally quantify the resilience gain over a restart scenario. This allows us to identify the scenarios in which dynamic adaptation is particularly useful, and what attributes contribute in its favour.

A. Measured Execution Time

To measure the execution time of the individual reconfiguration services, we implement a small IEC 61499 application, and perform a minor adaptation. The application behavior is simplistic and irrelevant to the measurements (Figure 4a). The adaptation we perform is the exchange of a single FB and the corresponding removing and adding of connections. After



(a) IEC 61499 Application

(b) Reconfiguration Control Application

Fig. 4. The test reconfiguration control application (4b) switches the FB E_SR_A in the application (4a) every two seconds using 12 reconfiguration operations.

TABLE I
STATISTICS OF THE MEASURED EXECUTION TIMES OF THE RECONFIGURATION FBS, AS PERFORMED BY THE TEST APPLICATION.

Function Block	Mean	Std	Min	Max
EC_START_ELEM	2.98 µs	1.38 µs	2.46 µs	21.28 µs
EC_STOP_ELEM	3.22 µs	1.54 µs	2.50 µs	25.45 µs
ST_CREATE_CONN	3.87 µs	1.46 µs	3.09 µs	25.09 µs
ST_CREATE_FB	11.34 µs	3.27 µs	9.20 µs	51.06 µs
ST_DEL_CONN	4.05 µs	1.38 µs	3.26 µs	24.48 µs
ST_DEL_FB	3.36 µs	1.15 µs	2.91 µs	19.19 µs

the adaptation is done, we revert the changes back to the initial configuration. The RCA consists of two sequences of each 12 operations and is displayed with the distinct phases in Figure 4b. Within each sequence, we first create the new FB, before we stop the old FB and the one connected to it to preserve consistency [6]. Then we delete the old connections, add the new connections, resume the FBs and delete the old FB. Every two seconds, the FB is switched and we measure the execution times of each FB within the RCA.

The measurement is performed on a Raspberry Pi 4B+ with Raspberry Pi OS with the PREEMPT_RT patch applied. The IEC 61499 application is modelled in 4diac and executed in 4diac FORTE [22] for one hour. The source code is modified to log relevant scheduling information, leading to 1800 data samples. All superfluous OS services and throttling are disabled, and the RT priority of 4diac FORTE is maximized. The results of this measurement are summarized in Table I, and the distributions are visualized in Figure 5. We can identify two groups: Most services are performed in around or under 4 µs, while the creation of a new FB requires on average 11.34 µs. The distributions show a long tail for longer execution times, which may be due to the execution platform, the non-real-time operations system, or the runtime environment. For our analysis, we stick to the mean value as the estimated execution time for each service. We do not advocate using our data for the sake of inferring execution times for safety-critical applications, since they depend critically on the hardware and software configuration, yet they suffice to support our analysis.

B. Estimated Adaptation Times

With the measurements of the execution times for each service, it is possible to estimate the necessary adaptation time

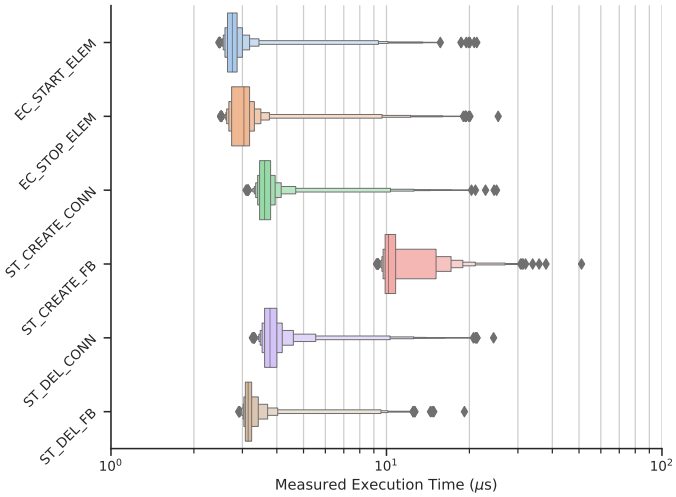


Fig. 5. The measured execution times of the reconfiguration FBs indicate a mean execution time of under 5 µs, except for ST_CREATE_FB, which takes around 11 µs. The outliers in the measurement over one hour are rare and have an insignificant influence on overall execution times.

for different scenarios. We assume four adaptation scenarios: A minor change, as seen in the previous section, where only minimal changes are performed, a moderate change of multiple components, and a major adaptation, which affects large portions of an application. Additionally, we consider a composite reconfiguration to represent a distributed scenario in which multiple devices are reconfigured. For the sake of simplicity, we ignore additional communication overhead in this scenario. For the moderate and major reconfigurations, we assume 10 and 100 added / removed FBs, respectively, and an average of three connections per FB. In practice, these numbers are easily achievable if hierarchical subapplications are modified.

We structure the adaptation into the five phases as described in Section IV-B. The resulting number of operations per phase and the corresponding estimated execution times are summarized in Table II. In our example scenarios, the number of operations scales linearly with the number of changed FBs. The estimated execution time behaves identically. The PRE and POST phases represent a significant overhead, yet they do not influence the real-time execution. While a minor adaptation only requires execution time in the order of µs, a moderate to composite scenario may require milliseconds. Ignoring the impact on the real-time execution (which is only affected by the RECONF duration), this adaptation is fast, but not instantaneous.

These results represent only the expected execution times. To reach the expected *adaptation* time, the utilization of the device must be taken into account, since the system will be busy with other tasks. Most of the operations will be performed with a low priority to prevent any disturbance of the real-time tasks. Consequently, for a device with 80% utilization, the adaptation time will be 5-times as long. This behavior is further analyzed in Figure 6, where the estimated adaptation times are plotted over the system utilization. At a utilization of 1.0, the adaptation is infeasible, since the additional load

TABLE II
OPERATIONS FOR EACH PHASE FOR FOUR ADAPTATION SCENARIOS, AND THE CORRESPONDING ESTIMATED EXECUTION TIME (MS).

	PRE	RINIT	RECONF	RDINIT	POST
Minor	$n = 36$	1	10	1	36
	0.23 ms	0.01 ms	0.04 ms	0.00 ms	0.14 ms
				$\sum$	**0.42 ms**
Moderate	360	10	100	10	360
	2.29 ms	0.11 ms	0.36 ms	0.03 ms	1.38 ms
				$\sum$	**4.17 ms**
Major	3600	100	1000	100	3600
	22.90 ms	1.13 ms	3.62 ms	0.34 ms	13.75 ms
				$\sum$	**41.73 ms**
Composite	36000	1000	10000	1000	36000
	228.96 ms	11.34 ms	36.16 ms	3.36 ms	137.52 ms
				$\sum$	**417.34 ms**

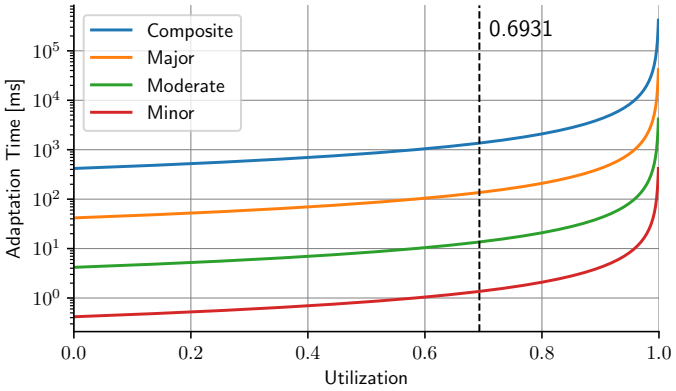


Fig. 6. The adaptation time depends on the system utilization. For realistic utilization levels, the adaptation will terminate within seconds.

would result in missed deadlines. The lower utilization bound for RM scheduling (0.6931) indicates a typical load that can be expected. This would allow a major adaptation to take place in about 100 ms, and a composite adaptation in one second. Since the utilization bound only considers real-time tasks, the actual utilization may be higher, and thus the adaptation time could be longer. Further communication overhead could lead to adaptation times in the order of seconds.

An important distinction is that the utilization of the system is decided at design time. While a high utilization is generally desirable to use the available resources efficiently, this may make dynamic adaptation infeasible or slow. More efficient scheduling paradigms, such as earliest deadline first, may allow even higher utilizations for real-time tasks, thus making dynamic adaptation even more difficult.

C. Resilience Scenarios

Using the aforementioned estimated adaptation times, we can now quantify the effect of dynamic adaptation on the system resilience. We focus first on the two cases of *survival* and *recovery*, an example of *prevention* will be analyzed as part of Figure 8.

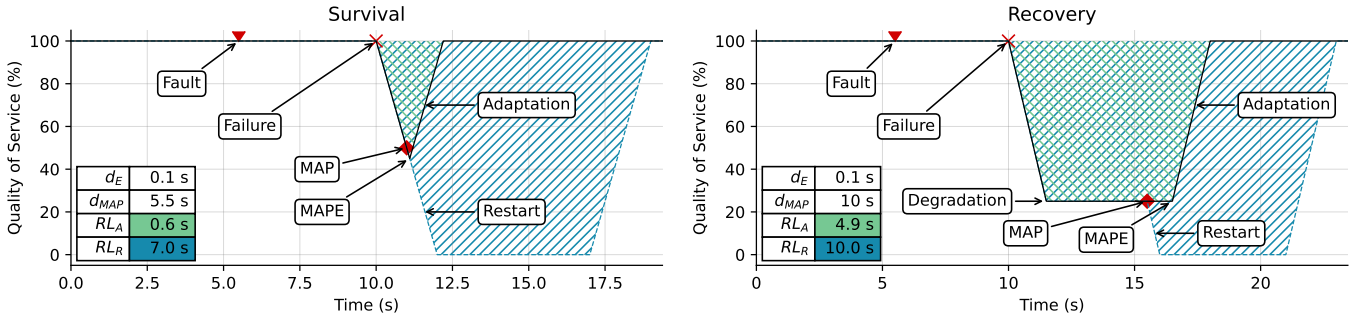
We base our analysis on a couple of assumptions. Degradation and recovery follow the resilience graph as introduced in Section III. A fully-available system has a Quality of Service (QoS) of 100 %, while a degraded system retains 25 %. During a restart, the system is disabled. The fault takes places after 5.5 s, the failure after 10 s. Degradation, recovery, and ramp-down/ramp-up require 2 s to transition from 100 % to 0 %. A restart takes 5 seconds. The values are inspired by realistic applications, however tailored to provide a meaningful analysis. A fault may remain dormant for days or weeks, and a restart may require a ramp-down in the order of hours to reach a safe state.

In Figure 7a, a survival scenario is displayed. The reaction takes place after 5.5 seconds (d_{MAP}), when the failure has happened and degradation has begun. The restart action quickly shuts down the system and performs the modification offline. Using a fast dynamic adaptation, the system can survive the failure, without reaching a degraded state or having to shut down. The major adaptation ($d_E = 0.1s$) can be performed nearly instantaneously. The resilience loss RL_A shows a loss of 0.6 s of production time, which is mostly caused by the degradation and ramp-up. The restart requires a significant loss of productivity ($RL_R = 7.0s$).

The recovery scenario in Figure 7b results from a delayed reaction time ($d_{\text{MAP}} = 10s$), which could be caused by a slow detection, or a complex decision making algorithm. The system reaches its degraded state, until finally a reaction is triggered. In the restart reaction, the system will quickly ramp down and perform a restart. In the adaptation scenario, the (in this case, complex) adaptation is triggered, which will cause the system to remain degraded until the adaptation is done. Finally, the recovery can begin. Similarly to the survival scenario, dynamic adaptation provides a significant resilience advantage over a restart. The loss RL_A of 4.9 seconds is still half the expected loss RL_R of a restart.

Further scenarios are sampled in Figure 8. There are four types of reaction speeds (early, fast, late, and delayed), and three levels of adaptation complexities are considered (Minor/Moderate, Major, and Composite). An early reaction coupled with a minor to major adaptation can, in this example, prevent an impact on the system, which results in a loss of 0, i.e. the system is perfectly resilient against this kind of fault/failure. The composite adaptation will lead to the manifestation of the failure, and a brief degradation. It must be noted that we assume a non-critical failure, that does not require an immediate shutdown. If the failure must be prevented at any cost, and the risk of its manifestation during the adaptation period is too large, then an emergency shutdown is necessary.

For the fast, late, and delayed reactions, various resilience losses can be observed. In all cases, the resilience loss of the adaptation (RL_A) is lower than the loss of the restart scenario (RL_R). This is evident from the fact that the adaptation time is always shorter than the restart time, which is a valid assumption. What is noteworthy is that once dynamic adaptation is available,



(a) Survival (b) Recovery

Fig. 7. In a survival scenario (7a), the adaptation takes place after the failure occurred, but before a degraded state is reached. The recovery scenario (7b) is a delayed reaction, once the degraded state has been reached. The resilience loss (RL) in the survival scenario is significantly lower, yet even in recovery, the adaptation provides a clear advantage over a restart.

the reaction time becomes the critical factor that determines the system resilience. Without dynamic adaptation, the reaction time is less important, since the restart will anyways lead to a loss. Consequently, the currently feasible dynamic adaptation mechanisms require and facilitate a shift of focus to monitoring, analysis, and planning phases.

D. Discussion

There are two key insights: First, the advantage of adaptation over a restart depends on how difficult a restart is. In our scenarios, we expect a restart to be feasible and reasonably fast, i.e. within five seconds. For many applications, this is highly optimistic and unrealistic. Restarting a PLC can require an extensive ramp-down phase to bring the system into a safe and known state, from which the system can be safely started. If, on the other hand, a restart is feasible within a short timeframe, e.g. because the system is stateless or the change does not affect the state, then the advantage of dynamic adaptation fades. We argue that most bugs, failures, or attacks will require more complicated modifications.

Second, once dynamic adaptation is feasible, available, and sufficiently fast, the main resilience gain can result from better monitoring, analysis, and planning methods. After all, an adaptation can not be faster than instantaneous. More importantly, dynamic adaptation facilitates further developments in monitoring, analysis, and planning, since it significantly lessens the burden of adaptation. A fast reaction that is implemented too late is similarly ineffective as a quickly implemented reaction that is detected too late. Yet the ability to quickly adapt together with a fast detection results in exceptional flexibility and agility.

Dynamic adaptation allows the implementation of any imaginable adaptation of the system. This allows the reaction to events that can not be anticipated, e.g. attacks or bugs. Many faults and failures, by contrast, can be anticipated, and a reaction can be arranged beforehand, or even directly implemented within the control application. This allows for a faster reaction without requiring a lengthy adaptation or a restart, yet additional resources must be reserved. The preparation of a reaction may also be coupled with dynamic adaptation to allow a fast implementation of complex reactions.

VI. Research Directions

Dynamic adaptation can be of great value to ICS. It leads to greater flexibility and agility of industrial control software. Yet, while the IEC 61499 generally allows it, it is rarely used. We believe that dynamic adaptation is a key feature that distinguishes the IEC 61499 from its alternatives. There remain plenty of open research directions:

Real-time Scheduling The IEC 61499 allows for more sophisticated scheduling algorithms than the traditional cyclic PLC execution. While [7] proposes RM scheduling using a PCP, this is not currently implemented in any existing IEC 61499 runtime environment, and the models do not contain the required information, e.g. rates or deadlines.

Distribution Distribution is a crucial dimension of future ICS, and it must be considered during adaptation. Further, shifting from monolithic to distributed systems simplifies the simultaneous switch towards concurrency and parallelism, which could be a great advantage, but is not currently used.

Monitoring, Analysis, Planning As previously stated, dynamic adaptation facilitates further developments in monitoring, analysis, and planning. The models of the IEC 61499 do not sufficiently reflect these phases yet: There are, for example, no satisfactory behavior models that could be monitored [23]. Automated planning is of great interest as well [24].

VII. Conclusion

Dynamic adaptation is a key component of self-adapting architectures. Within the MAPE-K model, eventually the necessary adaptations must be implemented, and this can be achieved through dynamic adaptation. In ICS, dynamic adaptation comes with additional requirements on consistency and timeliness, which can be satisfied [6, 7]. Yet, the feasibility of dynamic adaptation using the state of the art in research was an open question. In this paper, we demonstrated the potential of dynamic adaptation in contrast to a restart of a controller. The results indicate that the vast majority of adaptations can be completed in under a second, while preserving consistency and real-time constraints. Moderate to major adaptations can be achieved in the order of milliseconds.

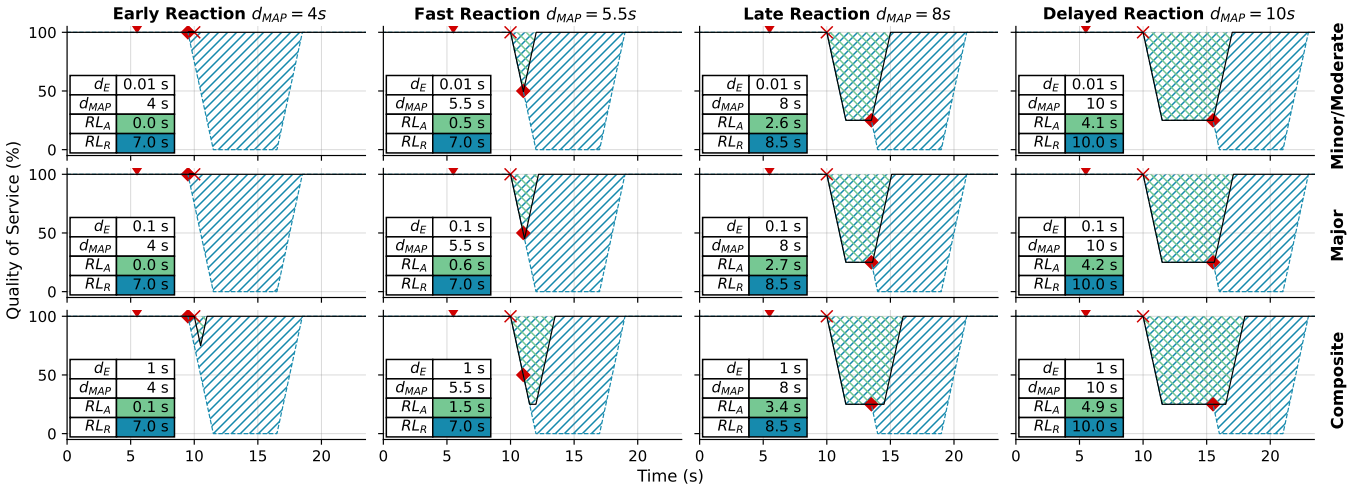


Fig. 8. Dynamic adaptation leads to a significantly smaller resilience loss RL compared to a traditional restart. If the detection and decision making (MAP, ♦) after the fault (▼) takes place before the failure (×), a degradation can be prevented (RL = 0). Otherwise, dynamic adaptation avoids lengthy downtimes and can recover much faster.

Our evaluation is based on conservative estimates, since most ICS can not be quickly stopped and restarted, but require long ramp-down and ramp-up phases to bring the system into a safe, initial state. Other systems can not be restarted at all. In comparison to systems based on the IEC 61131-3, the fractional state of the IEC 61499 facilitates surgical interventions where it is needed, allowing exceptionally fast adaptation of large applications. The possibility of dynamic adaptation shifts the focus towards the earlier phases of the MAPE-K model, most importantly Monitoring, Analysis, and Planning. It enables ICS to be frequently and quickly modified, thus allowing and demanding faster and more sophisticated detection and decision-making mechanisms. We believe that dynamic adaptation can lead to resilient, flexible, and agile ICS.

References

[1] J O Kephart and D M Chess. "The vision of autonomic computing". In: *Computer* 36.1 (Jan. 2003), pp. 41–50.

[2] Omid Gheibi, Danny Weyns, and Federico Quin. "Applying Machine Learning in Self-adaptive Systems: A Systematic Literature Review". In: *ACM Trans. Auton. Adapt. Syst.* 15.3 (Aug. 2021), pp. 1–37.

[3] Hansong Xu, Wei Yu, David Griffith, and Nada Golmie. "A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective". In: *IEEE Access* 6 (2018), pp. 78238–78259.

[4] Hoda A ElMaraghy. "Flexible and reconfigurable manufacturing systems paradigms". In: *International Journal of Flexible Manufacturing Systems* 17.4 (Oct. 2005), pp. 261–276.

[5] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. "Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation". In: *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, May 2015, pp. 13–23.

[6] Laurin Prenzel and Sebastian Steinhorst. "Automated Dependency Resolution for Dynamic Reconfiguration of IEC 61499". In: *International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sept. 2021.

[7] Laurin Prenzel, Simon Hofmann, and Sebastian Steinhorst. "Real-time Dynamic Reconfiguration for IEC 61499". In: *International Conference on Industrial Cyber-Physical Systems*. IEEE, 2022.

[8] A Avizienis, J-C Laprie, B Randell, and C Landwehr. "Basic concepts and taxonomy of dependable and secure computing". In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (Jan. 2004), pp. 11–33.

[9] Israel Koren and C Mani Krishna. *Fault-Tolerant Systems*. en. Morgan Kaufmann, Sept. 2020.

[10] Torben Stolte et al. "A Taxonomy to Unify Fault Tolerance Regimes for Automotive Systems: Defining Fail-Operational, Fail-Degraded, and Fail-Safe". In: *Transactions on Intelligent Vehicles* (2021).

[11] Simon Dobson, Roy Sterritt, Paddy Nixon, and Mike Hinchey. "Fulfilling the Vision of Autonomic Computing". In: *Computer* 43.1 (Jan. 2010), pp. 35–41.

[12] Danny Weyns et al. "On Patterns for Decentralized Control in Self-Adaptive Systems". In: *Software Engineering for Self-Adaptive Systems II*. Ed. by Rogério de Lemos, Holger Giese, Hausi A Müller, and Mary Shaw. Vol. 7475 LNCS. Springer, 2013, pp. 76–107.

[13] Laurin Prenzel and Sebastian Steinhorst. "Decentralized Autonomous Architecture for Resilient Cyber-Physical Production Systems". In: *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. Grenoble, France: IEEE, 2021.

[14] Johannes Iber, Tobias Rauter, Michael Krisper, and Christian Kreiner. "The potential of self-adaptive software systems in industrial control systems". In: *European Conference on Systems, Software and Services Process Improvement, EuroSPI 2017*. Vol. 748. Springer, Cham, 2017.

[15] Robert W Brennan et al. "Developments in dynamic and intelligent reconfiguration of industrial automation". In: *Computers in Industry* 59.6 (Aug. 2008), pp. 533–547.

[16] Alois Zoitl. *Real-time Execution for IEC 61499*. en. Instrumentation, Systems, and Automation Society, 2009.

[17] Seyedmohsen Hosseini, Kash Barker, and Jose E Ramirez-Marquez. "A review of definitions and measures of system resilience". In: *Reliability Engineering & System Safety* 145 (Jan. 2016), pp. 47–61.

[18] Erik Hollnagel, David D Woods, and Nancy Leveson. *Resilience Engineering: Concepts and Precepts*. Ashgate Publishing, 2006.

[19] Devanandham Henry and Jose Emmanuel Ramirez-Marquez. "Generic metrics and quantitative approaches for system resilience as a function of time". In: *Reliability Engineering & System Safety* 99 (Mar. 2012).

[20] Michel Bruneau et al. "A Framework to Quantitatively Assess and Enhance the Seismic Resilience of Communities". In: *Earthquake Spectra* 19.4 (Nov. 2003), pp. 733–752.

[21] Christoph Sünder, Valeriy Vyatkin, and Alois Zoitl. "Formal Verification of Downtimeless System Evolution in Embedded Automation Controllers". In: *ACM Transactions on Embedded Computing Systems* 12.1 (Jan. 2013), pp. 1–17.

[22] Alois Zoitl, Thomas Strasser, and Antonio Valentini. "Open source initiatives as basis for the establishment of new technologies in industrial automation: 4DIAC a case study". In: *International Symposium on Industrial Electronics*. IEEE, 2010.

[23] Bianca Wiesmayr and Alois Zoitl. "Requirements for a dynamic interface model of IEC 61499 Function Blocks". In: *International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. IEEE, 2020.

[24] Tarik Terzimehic, Sebastian Voss, and Monika Wenger. "Using Design Space Exploration to Calculate Deployment Configurations of IEC 61499-based Systems". In: *International Conference on Automation Science and Engineering (CASE)*. IEEE, 2018.