Safety and dependability analysis of self-adaptive systems

Matthias Güdemann, Frank Ortmeier, Wolfgang Reif

Angaben zur Veröffentlichung / Publication details:

Güdemann, Matthias, Frank Ortmeier, and Wolfgang Reif. 2006. "Safety and dependability analysis of self-adaptive systems." In *Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (isola 2006), 15-19 November 2006, Paphos, Cyprus*, edited by Tiziana Margaria and Bernhard Steffen, 177–84. Piscataway, NJ: IEEE. https://doi.org/10.1109/isola.2006.38.

Safety and Dependability Analysis of Self-Adaptive Systems

Matthias Güdemann, Frank Ortmeier and Wolfgang Reif
Lehrstuhl für Softwaretechnik und Programmiersprachen,
Universität Augsburg, Universitätsstrasse 14, D-86135 Augsburg
{guedemann, ortmeier, reif}@informatik.uni-augsburg.de

***Abstract*— In this paper we present a technique for safety analysis of self-adaptive systems with formal methods. Self-adaptive systems are characterized by the ability to dynamically (self-)adapt and reorganize. The aim of this approach is to make the systems more dependable. But in general it is unclear how big the benefit is - compared to a traditional design.**

We propose a dependability analysis based on the results of safety analysis to measure the quality of self-x capabilities of an adaptive system with formal methods. This is important for unbiased and evidence-based decision making in early design phases. To illustrate the results we show the application of the method to a case study from the domain of production automation.

I. Introduction

Many modern systems have a very static nature. This means they can not very well adapt to changes in the requirements nor to failure of individual components. A possible way to make such systems dependable and failure tolerant is to build redundancy in many components. This can be very costly and is often impossible due to space requirements. New approaches like Organic Computing [1] or IBM's autonomic computing initiative [2] try to design systems from the beginning in such a way, that they can dynamically self-adapt to their environment. These systems use the "let the system go" paradigm, i.e. not every special case is pre-programmed and the system has some degree of freedom to make its own choices.

Open questions are how to conduct safety analysis on these systems, how big the gain (of adaption compared to conventional design) is and how it can be measured. Standard methods for reliability analysis like FMEA [3], FTA [4] and DCCA [5] are not directly applicable, because they try to find only cause-consequence relationships between component failures and system failures. In contrast, self-healing of an adaptive system makes it possible, that the system autonomously recovers from a hazardous state. But for evaluation of adaptive systems for safety critical applications, formal methods are crucial [6].

We show how a formal technique from the domain of safety analysis can be extended to adaptive systems and be used to measure the gain in dependability. We illustrate the method on a case study from production automation. As we regard dependability as the capability to adapt to failures we also use the term adaptability throughout this paper for the same purpose.

The case study is presented in Sect. II. In Sect. III we give a brief introduction to safety analysis and explain the theory of measuring self-x with it. Metrics for dependability are shown in Sect. IV and Sect. VI concludes the paper.

II. Case Study

The case study describes an automated production cell with three robots, which are connected with autonomous transportation units. Every robot can accomplish three tasks: drilling a hole in a workpiece, inserting a screw into this hole and tightening the screw. These tasks are accomplished with three different tools that can be switched. Every workpiece must be processed by all three tools in the given order (drill, insert and tighten = DIT). Workpieces are transported between the robots by autonomous transportation units (carts). Changing the tool of a robot requires some time. Therefore the standard configuration of the system is that the three tasks are spread out between the three robots and the carts transfer workpiece accordingly. This situation is shown in fig. 1.

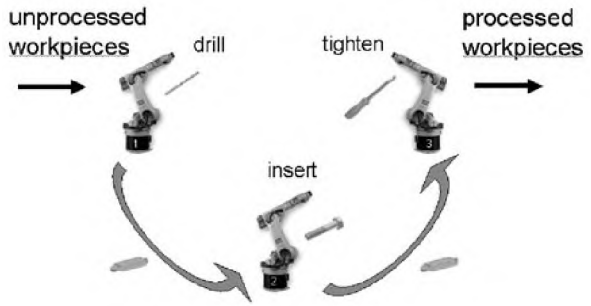

Fig. 1. Valid configuration of robot cell

We examine the case when one or more tools break and the current configuration allows no more correct DIT processing of the incoming workpieces. fig. 2 If the drill of one robot breaks then DIT processing is no more possible, as no other robot is configured to drill. A non-adaptive production cell would now come to a standstill and wait for maintenance.

However, it is obvious, that this is not a real hazard as the robots have three tools and can switch to another tool if one breaks. So it should be possible for the adaptive system to detect this situation and reconfigure itself in such a way, that DIT processing becomes possible again. This implies that at

least one other robot also has to switch its tool, so that all three tools are available again.

This can be resolved like shown in fig. 3. For this error resolution, not only the assignment of the tasks to the robots must be changed, but also the routes of the carts. If only the tools were switched, the processing of all tasks would be possible, but not in the correct order. The reconfiguration is triggered and executed by an external observer who then assigns new tasks to the robots and carts.

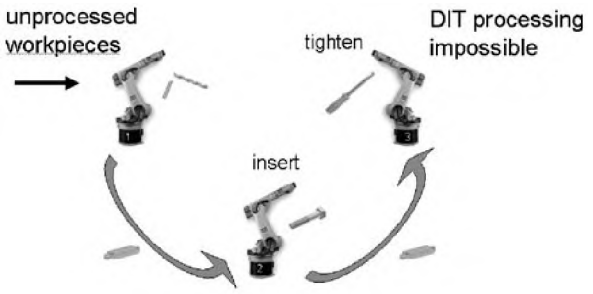

Fig. 2. Temporary hazard (broken drill)

If an adaptive system can achieve this reconfiguration autonomously, then it can be seen as superior to traditional system. The interesting question is now: "How can this improvement be measured?" The presented example only shows one reconfiguration for one error. What will happen if several errors occur over time, triggering several reconfiguration steps? It is even possible, that during processing unused tools of the robots are repaired by maintenance. If all these effects are taken into account it soon becomes very hard to predict, if the system can recover again or not.

To formally analyze this system we first formulated the correct functioning of the system as the invariant predicate "The system can process a workpiece in DIT order". Violation of this invariant is detected by the observer who then restores the invariant if possible. Using this method, one can see reconfiguration abstractly as restoration of invariants.

A. Formal Model

The formal model was implemented as transition system in the SMV model checker [7] which is very powerful and used

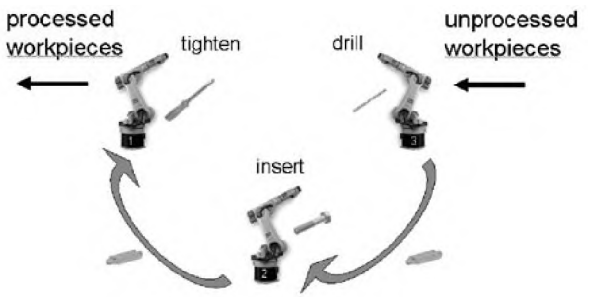

Fig. 3. Reconfigured robot cell

for analysis of adaptive systems [8]. We did not implement a reconfiguration algorithm in the model but only specified it in a top-down way, as restoration of the functional invariant "the system has DIT capability and the carts are correctly configured", using our "restore-invariant" approach [9].

Due to space restrictions not the whole transition system is shown for the automata. The respective transition preconditions are explained in the text. Dashed lines indicate the effect of an interrupting reconfiguration. If used it confines reconfiguration from normal functioning.

B. Transition Systems

The production cell was implemented as product automaton of the transition systems for the robots, the workpieces, the reconfiguration control and the transportation carts. Due to space restrictions only the details necessary for the next sections are presented. A more detailed description can be found in [9].

The Reconfigurator is modeled by the $Control$ automaton. It assigns tasks to the robots and the carts sequentially. Which tasks are assigned is chosen indeterministically, the assignments are guaranteed to be a valid configuration as explained in Sect. II-D. After Reconfiguration, the Control enters the $EndReconf$ state and immediately afterwards the $None$ idle state that is only left if another reconfiguration is necessary due to invariant violation.

The robot automata have states that are associated with the task they have to fulfill. For each task there is a waiting state, working state and a done state. The robots also have a $Reconf$ state so that they can have another task assigned. The carts are modeled as product automaton of three transition systems. The first C_j^{conf} models the configuration of the cart (either $Reconf$ or a task), C_j^{pos} models the position of the cart in the cell and C_j^{state} models the current state of the cart. The workpieces are also modeled as product automaton. The position is modeled by WP_j^{pos} and the current state by WP_i^{state}.

1) Control transition system: The Control does the reconfiguration of the production cell. Its transition system is shown in fig. 4. It waits in state $Reconf$ until all robots and carts are in their respective reconfiguration states. When this has happened the control enters the state $Initialize$. From this state on the Control enters one of the states of the $Robot1Conf$ multi-state, then one of the $Robot2Conf$ states and finally one of the $Robot3Conf$ states. Which one of the states is entered, decides which task is assigned to the corresponding robot, i.e. when the control enters state $robotD$ in the multi-state $Robot1Conf$, then robot 1 is assigned to use its drill. Which task is assigned to which robot is chosen indeterministically. The assignment of the routes to the carts is done analogously in the $Cart1Conf$ and $Cart2Conf$ multi-states. Whether this assignment allows correct processing of workpieces is assured by the specification of the reconfiguration algorithm explained in Sect. II-D.

2) Robot transition systems: For the robot we only describe the part of the transition system responsible for the drill task. For the two other tasks, equivalent states exist starting from

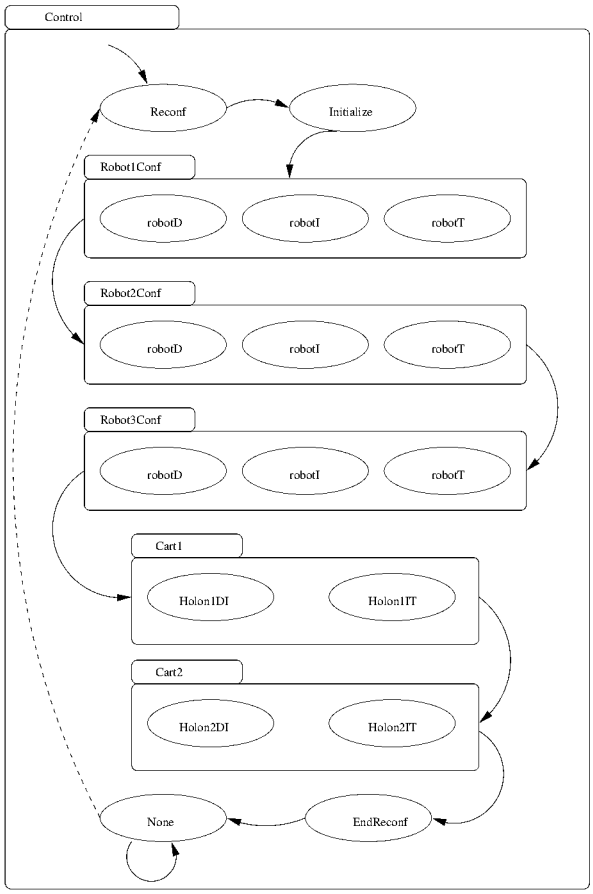

Fig. 4. Control transition system

the initial state $Reconf$. This state is left when the Control assigns a new task to the corresponding robot. The succeeding states are either $readyD$, $readyI$ or $readyT$ for the respective tasks.

When the robot is in $readyD$ state it waits for a new workpiece to arrive. When this happens it enters state $busyD$. If the workpiece has already been processed with the tool the robot uses, it enters directly $doneD$, simulating passing through of the already processed workpiece. After $busyD$ the $doneD$ state is entered indicating that the workpiece processing is complete and the robot waits for a cart that fetches the workpiece. When this happens, the robot enters $readyD$ again. Whenever a new reconfiguration is initiated by the Control, then a robot leaves its current state and reenters $Reconf$.

We could add a different number of *busy* states for each tool, to simulate the differing durations of each task. This is not necessary for functional properties, but could be used for temporal propositions, like: "A workpiece is always processed in less than k timesteps." This can be integrated in our model by using the RTCTL [10] syntactic extension of CTL.

To prove functional properties about correct production of the cell and for correct sequence of transportation, formal

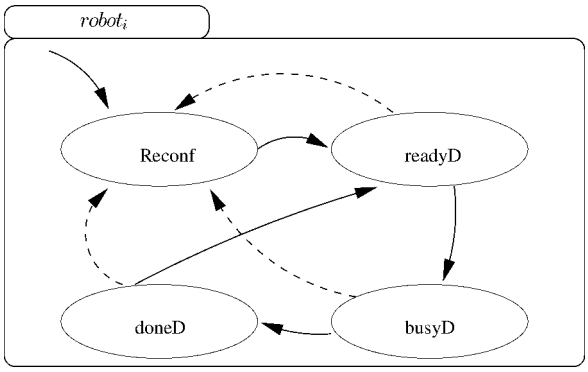

Fig. 5. Robot transition system

models of the autonomous carts and the workpieces are also needed. The full model also consists of the parallel composition of automata for the position of workpieces and their state (drilled, inserted, tightened) as well as the position, the state (idle, loaded, heading back) and the configuration of the autonomous carts.

C. Predicates

We use the notion $A = s$ as abbreviation for the predicate "automaton A is in state s". For the formal model we define the predicates R_j^a with $j \in \{1,2,3\}$ and $a \in \{d,i,t\}$. These variables are true if robot j has been assigned task a by checking whether the corresponding automaton is in one of the states corresponding to task a. Together with the configuration automata of the carts we define:

$$\begin{aligned}
conf &:= (R_1^d \vee R_1^i \vee R_1^t) \wedge (R_2^d \vee R_2^i \vee R_2^t) \\
&\wedge (R_3^d \vee R_3^i \vee R_3^t) \\
&\wedge (C_1^{conf} \neq Reconf \wedge C_2^{conf} \neq Reconf) \\
ditCap &:= \bigwedge_{a\in\{d,i,t\}} \Big(\bigvee_{j\in\{1,2,3\}} R_j^a \wedge \Big(\bigwedge_{k\in\{1,2,3\}\setminus j} \neg R_k^a \Big) \Big) \\
cartCap &:= (C_1^{conf} \neq C_2^{conf}) \wedge (C_1^{conf} \neq Reconf) \\
&\wedge (C_2^{conf} \neq Reconf)
\end{aligned}$$

This means that $conf$ holds if all robots and carts have a task assigned.The variable $ditCap$ is true if the assignment of tasks to robots includes all three tasks. As the variables R_j^a are defined via the states of the robot automata, the formula $R_j^a \rightarrow \bigwedge_{b\in\{d,i,t\}\setminus a} \neg R_j^b$ always holds, i.e. every robot can only have one task assigned.

To model broken tools as failures in the model of the production cell we define transient failure automata as explained in Sect. III-A for all tools of all robots. These failure automata are called $fails_j^a$ with $j \in \{1,2,3\}$ and $a \in \{d,i,t\}$. Using these automata we define additional boolean variables:

$$\begin{aligned}
ditFailure_j &:= \bigvee_{a\in\{d,i,t\}} (R_j^a \wedge (fails_j^a = yes)) \\
ditFailure &:= \bigvee_{j\in\{1,2,3\}} ditFailure_j
\end{aligned}$$

This means that $ditFailure_j$ holds if robot j has been assigned a task it cannot perform as the corresponding tool is broken and

The variable $ditFailure$ indicates that one or more robots have been assigned a task that is impossible at the moment. Whenever the external Control detects that $ditFailure$ holds, then a reconfiguration is triggered.

For proving functional properties and specifying a correct reconfiguration algorithm we also need the predicate $ditPossible$ that holds if a correct configuration is still theoretically possible. We used the disjunction of all correct robot configurations for this. It is important to mention, that this is not needed in the model itself but only to specify the reconfiguration algorithm and to prove functional correctness. That means that $ditPossible$ may also be defined in another way for example to model *graceful degradation* adaption.

D. Specification of Reconfiguration

As mentioned earlier, the reconfiguration algorithm is not implemented directly, but specified as temporal logic formulas. For this specification we used LTL (Linear Time Temporal Logic) [11]. LTL is used for these specifications as SMV allows only LTL in assumed properties. The two specifications are as follows:

$$\begin{aligned} confDIT &:= \mathbf{G}\,(conf \rightarrow ditCap \wedge cartCap) \\ confCorrect &:= \mathbf{G}\,((Control = EndReconf) \rightarrow \\ &\qquad \mathbf{X}\,(ditPossible \rightarrow \neg ditFailure)) \end{aligned}$$

That is, $confDIT$ specifies that every reconfiguration results in a sensible configuration of the production cell according to the specification that every tool must be available and the carts have distinct routes assigned between the right robots. The second property $confCorrect$ specifies that whenever $Control = EndReconf$, i.e. a reconfiguration has just been finished, then in the next step $ditFailure$ is false as long as $ditPossible$ is true. This means that a reconfiguration is always successful ($\neg ditFailure$) as long as a correct reconfiguration is still possible despite potential occurrences of failures. All functional properties are proven under the assumption that $confDIT$ and $confCorrect$ hold. These two properties specify a reconfiguration algorithm without explicit implementation. We call this procedure the "restore-invariant" approach. Whenever the invariant $\neg ditFailure$ is violated, $confDIT$ and $confCorrect$ restore it as long as still possible. This separates modeling of the production cell from a reconfiguration algorithm. Any explicit reconfiguration algorithm that fulfills these two propositions keeps the cell functionally correct.

Assuming the properties $confDIT$ and $confCorrect$ it was possible to proof the functional property of the production cell that "workpieces that leave the cell are processed with all tools" and that "workpieces are never processed in wrong order". A more detailed analysis can be found in [9].

III. Deductive Cause Consequence Analysis

Deductive Cause Consequence Analysis (DCCA) is used to find the cause - consequence relationship between failures and hazards, i.e. which combination of failures can be a cause for a given hazard. The formalization of DCCA is done with Computational Tree Logic (CTL) [11]. We use finite automata as system models. The use of CTL and finite automata allows to use powerful model checkers like SMV [7] to verify the proof obligations.

In the following we assume that a list of hazards on system level and a list of possible basic component failure modes is given. Both data may be collected by other safety analysis techniques like failure-sensitive specification, see [12] or HazOp, see [13]. We assume that system hazards H and primary failures δ are described by predicate logic formulas. This is true for many practical problems. If the system hazard cannot be described by a predicate logic formula directly, then often an observer automaton may be implemented such that whenever the automaton is in an accepting state, the hazard has occurred before [14]. We call the set of all failure predicates Δ and assume there is only a finite number of failures.

A. Failure/Hazard Automata

For formal safety analysis failure modes must be explicitly modeled. We divide the modeling into two steps. First we model the occurrence pattern of the failure mode and second we model the failure mode itself. By "occurrence pattern" we understand how and when the failure mode occurs. For example does the failure mode occur indeterministically (like packet loss in IP traffic) or does it occur once and forever (like a broken switch) or does it occur only during certain time intervals (like until the next maintenance). To model this we use failure automata. Fig. 6 shows two such failure automata.

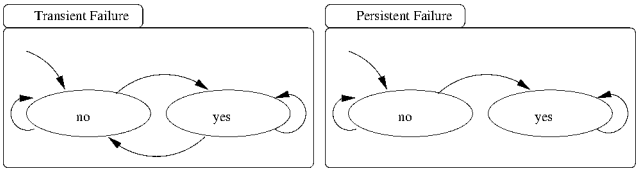

Fig. 6. Failure automata for transient and persistent failures

The left automaton models a transient failure which can indeterministically occur and disappear. The right one models a persistent failure, which happens once and stays forever. Failure predicates δ are then defined as "failure automaton for failure mode δ in state *yes*". For readability the symbol δ is used for both the predicate and the automaton describing the occurrence pattern.

The second step is to model the direct effects of failure modes. This is usually done by adding transitions to the model with conditions of the form $\varphi \wedge \delta$. This means these additional transitions – which reflect erroneous behavior – may only be taken, when a failure automaton is in state *yes* i.e. when a failure occurs.

A similar approach may be used to define predicates for system hazards. If the system hazard cannot be described by a predicate logic formula directly, then often an observer automaton may be implemented such that whenever the automaton is in an accepting state, the hazard has occurred before [14]. This allows to describe the hazard as predicate logic formula on the states of the observer automaton. However in practical applications hazards may usually be described by predicate logic formulas.

B. DCCA for Non-adaptive Systems

The next step is to define a temporal logic property which says, whether a certain combination of failure modes may lead to the hazard or not. This property is called *criticality* of a set of failure modes.

Definition *1:* Critical set / minimal critical set

For a system SYS and a set of failure modes Δ a subset of component failures $\Gamma \subseteq \Delta$ is called critical for a system hazard if

$$SYS \models \mathbf{E}\,(\overline{\lambda}\ \mathbf{until}\ H) \ \ where\ \overline{\lambda} := \bigwedge_{\delta \in (\Delta \setminus \Gamma)} \neg\,\delta$$

We call Γ a minimal critical set if Γ is critical and no proper subset of Γ is critical.

Here, $\mathbf{E}\,(\varphi\,\mathbf{until}\,\psi)$ denotes the existential CTL-UNTIL-operator. It means there exists a path in the model, such that φ holds until the property ψ holds. The property *critical set* translates into natural language as follows: there exists a path such that the system hazard occurs without the previous occurrence of any failures except those which are in the critical set. In other words this means, it is possible that the systems fails, if only the component failures in the critical set occur. Intuitively, criticality is not sufficient to define a cause-consequence relationship. It is possible that a critical set includes failure modes, which have nothing to do with the hazard.

Therefore, the notion *minimal critical set* also requires that no proper subset of it is critical. Minimal critical sets really describe what one would expect for a cause-consequence relationship in safety analysis to hold: the causes may - but not necessarily - lead to the consequence and second all causes are necessary to allow the consequence to happen. A DCCA is called complete, if for all $\Gamma \in \mathcal{P}(\Delta)$ it is decided, if Γ is a minimal critical set or not. It can then be shown, that for a complete DCCA the hazard cannot occur unless previously at least one of the minimal cut sets has taken place [5]. Therefore minimal critical sets generalize the notion of minimal cut sets of FTA [4], [15].

So the goal of DCCA is to find minimal critical sets of failure modes. Testing all sets by brute force would require an effort exponential in the number of failure modes. However, DCCA may be used to formally verify the results of informal safety analysis techniques. This reduces the effort of DCCA a lot, because the informal techniques often yield good "initial guesses" for solutions. Note that the property critical is monotone with respect to set inclusion i.e. $\forall \Gamma_1, \Gamma_2 \subseteq \Delta : \Gamma_1 \subseteq \Gamma_2 \Rightarrow (\Gamma_1$ is critical set $\Rightarrow \Gamma_2$ is critical set). This helps to reduce proof efforts a lot.

C. Analyzing Adaptive Systems

An adaptive system cannot directly be analyzed with DCCA (or another traditional safety analysis technique). This is because all these methods try to find a cause-consequence relationship between component failures and system failure. Adaptive systems are different, because if the hazard occurs and the reconfiguration succeeds, then the system will come back into working mode some time later. So the occurrence of the hazard is only "critical" if the system cannot repair itself anymore. Intuitively the hazard is only critical, if the system cannot repair itself with reconfiguration. In other words: if the self-x capabilities may not compensate the failures any more. The problem can be solved if the definition of the notion of criticality is extended:

Definition *2:* Critical set / minimal critical set for adaptive systems

For a system SYS and a set of failure modes Δ a subset of component failures $\Gamma \subseteq \Delta$ is called critical for a system hazard if

$$SYS \models \mathbf{E}\,(\overline{\lambda}\ \mathbf{until}\ (\,\mathbf{EG}\,(\,H \wedge \overline{\lambda}))) \ \ where \overline{\lambda} := \bigwedge_{\delta \in (\Delta \setminus \Gamma)} \neg\delta$$

We call Γ a minimal critical set if Γ is critical and no proper subset of Γ is critical.

The formula in Def. 2 states that a set of failure modes Γ is critical if there exists a trace such that only these failure modes can lead to permanent system failure. The last part in the formula "$\wedge\overline{\lambda}$)" is necessary, to make sure really the failures of Γ are the reason for the permanent system failure. With this extension a similar completeness theorem can be formulated.

Just like with traditional DCCA (see [5]), the *criticality* property of a set of failures is also monotonic for the notion of *critical sets* as in Def. 2, i.e. $\forall\, \Gamma_1, \Gamma_2 \subseteq \Delta : \Gamma_1 \subseteq \Gamma_2 \Rightarrow$ (Γ_1 is critical set $\Rightarrow \Gamma_2$ is critical set).

For the following completeness theorem of Adaptive-DCCA we use CTL* temporal logic [11], as the proof is more straightforward than with only CTL formulae. Nevertheless, to use Adaptive-DCCA for safety analysis only the formula of Def. 2 is needed which is expressible in CTL and thus analyzable for a model checker like SMV.

Theorem *1:* Adaptive-DCCA completeness theorem

For a complete DCCA for a model of the system $\mathcal{FM}$, a set of failure modes Δ and an hazard H the following formula holds:

$$\mathcal{FM} \models \mathbf{A}\,((\bigwedge_{\Gamma \in mcss(\mathcal{FM},\Delta,H)} \neg \bigwedge_{\delta_j \in \Gamma} \mathbf{F}\,\delta_j) \rightarrow \neg\,\mathbf{FG}\,H)$$

This formula basically states, that on all those traces, where no minimal critical set occurs, the hazard can always be repaired by the system again.

Proof:

Assume:

$$\mathcal{FM} \not\models \mathbf{A}\,(((\bigwedge_{\Gamma \in mcss(\mathcal{FM},\Delta,H)} \neg \bigwedge_{\delta\in\Gamma} \mathbf{F}\,\delta) \rightarrow \neg\,\mathbf{FG}\,H) \quad (1)$$

$\Leftrightarrow$ there exists $\sigma = (\sigma_0, \ldots) \in \mathcal{FM}$:

$$\mathcal{FM}, \sigma \models (\bigwedge_{\Gamma \in mcss(\mathcal{FM},\Delta,H)} \neg \bigwedge_{\delta\in\Gamma} \mathbf{F}\,\delta) \quad (2)$$

and $\mathcal{FM}, \sigma \models \mathbf{FG}\,H$ (3)

Since the set Δ is finite, there is a state σ_i on σ such that all component failures δ that will ever happen have occurred at least once. Therefore we can choose i such that

$$\forall \delta \in \Delta : \ (\forall j < i : \ \mathcal{FM}, \sigma_j \models \neg\delta) \rightarrow \mathcal{FM}, \sigma_i \models \mathbf{G}\,\neg\delta \quad (4)$$

Let $\Gamma := \{\delta \in \Delta \mid \exists\, j < i \ : \mathcal{FM}, \sigma_j \models \delta\}$ (5)

$\overset{(3),(5)}{\Rightarrow}$ $\mathcal{FM}, \sigma \models (\overline{\Gamma}\,\mathbf{until}\ \mathbf{EG}\,(H \wedge \overline{\Gamma}))$

$\Leftrightarrow$ Γ is critical set (Def. 2)

$\Rightarrow$ $\exists\, \Gamma_0 \subseteq \Gamma :$ Γ_0 is minimal critical set

therefore $\Gamma_0 \in mcss(\mathcal{FM}, \Delta, H)$

$\overset{(5)}{\Rightarrow}$ $\mathcal{FM}, \sigma \models \bigwedge_{\delta\in\Gamma} \mathbf{F}\,\delta$ (6)

$\Rightarrow$ $\mathcal{FM}, \sigma \models \bigwedge_{\delta\in\Gamma_0} \mathbf{F}\,\delta$, as $\Gamma_0 \subseteq \Gamma$

$\Rightarrow$ $\mathcal{FM}, \sigma \not\models ((\bigwedge_{\Gamma \in mcss(\mathcal{FM},\Delta,H)} \neg \bigwedge_{\delta\in\Gamma} \mathbf{F}\,\delta)$

$\Rightarrow$ $\lightning$ to (2)

■

IV. Measuring Dependability

The results from DCCA extended to adaptive systems can be used as measurement of adaptability. As we use adaption to failures, we see this also as a measurement of dependability. We propose several different metrics and present the results of applying DCCA to our case study of the adaptive production cell.

A. DCCA and Adaptability Metrics

To analyze an adaptive system with DCCA one must integrate failures into the model, by specifying the corresponding failure automata as described in Sect. III-A. After failure integration, the minimal critical sets of the system have to be computed with the help of the equation in Def. 2. It is advisable to start with small sets or guessed sets, from FMEA for example. As criticality is monotonic (see Sect. III-C), sets that have a proper critical subset do not have to be analyzed. Failures are integrated into model and the minimal critical sets are computed as described in Sect. III. This procedure is conducted for both the conventional and the adaptive system, i.e. we measure adaptability and dependability in relative terms. Because of this, the hazard H to which the minimal critical sets are computed must be the same for both systems and for the failure modes Γ of the conventional system and the failure modes Γ' of the adaptive system we assume $\Gamma \subseteq \Gamma'$, i.e. the extension to self-x properties might introduce new failure modes, but the ones from the original system may also occur.

1) Qualitative Metrics: The resulting minimal critical sets can be used for *qualitative* analysis, by comparing the size of the resulting sets as bigger sets mean better reliability. Only when all failures in a minimal critical set occur, then it can lead to a hazard, i.e. the bigger the set, the better the dependability of the system. Our metrics rely on the quotient of measures of the adaptive system and the conventional system, i.e. may be undefined. This can only happen if a system is functionally incorrect and therefore has an empty minimal critical set. As we basically measure the factors of increase in dependability, Therefore we assume functional correctness of the systems, i.e. at least one minimal critical set of size equal to or greater than one.

Definition *3:* Qualitative minimal metric / Qualitative average metric

The Qualitative minimal metric *for an adaptive system modeled by $\mathcal{FM}_{adaptive}$ with a set of failure modes Δ and a corresponding non-adaptive system modeled by $\mathcal{FM}_{conv}$ with a set of failure modes Δ' and a hazard H is defined as:*

$$d_{qual}^{min} := \frac{\min_{\Gamma \in mcss(\mathcal{FM}_{adaptive},\Delta',H)} |\Gamma|}{\min_{\Gamma' \in mcss(\mathcal{FM}_{conv},\Delta,H)} |\Gamma'|} \quad (7)$$

The Qualitative average metric *is defined as:*

$$d_{qual}^{avg} := \frac{avg_{adaptive}}{avg_{conv}} \quad (8)$$

with $avg_i := \frac{\sum_{\Gamma \in mcss(\mathcal{FM}_i,\Delta,H)} |\Gamma|}{|mcss(\mathcal{FM}_i,\Delta,H)|}$

The *Qualitative minimal metric* compares the sizes of the smallest minimal critical sets of both the conventional and the adaptive system. The size of this set is the minimal number of failures that must be present until a hazard may occur. is rather coarse but is helpful during development if all failures have a very low probability of occurrence. The *Qualitative average metric* determines the ratio of the average size of the minimal critical sets of the adaptive and the conventional model, therefore incorporating all found minimal critical sets. This metric does not only take the size of the critical sets in account, but also the total number of critical sets which may be a lot higher in the adaptive model.

A very desirable goal for building an adaptive system is to make as many minimal critical sets of the conventional system as possible non-critical for the adaptive system by adding at least one failure mode to each. That means that at least one additional failure must appear until the hazard H may occur. The *Qualitative inclusion metric* is based on this idea.

Definition *4:* Qualitative inclusion metric

The Qualitative inclusion metric *for an adaptive system modeled by $\mathcal{FM}_{adaptive}$ with a set of failure modes Δ and a corresponding non-adaptive system modeled by $\mathcal{FM}_{conv}$ with a set of failure modes Δ' and a hazard H is defined as:*

$$d_{qual}^{incl} := \begin{cases} \frac{|A|}{|B|} & \text{if } \forall \Gamma' \in D : \exists \Gamma \in C : \Gamma \subseteq \Gamma' \\ -1 & \text{otherwise} \end{cases} \quad (9)$$

$$\text{with} \quad A := \{\Gamma' | \Gamma' \subseteq \Gamma \wedge \Gamma' \in C \wedge \Gamma \in D \wedge \Gamma' \neq \Gamma\}$$
$$B := \{\Gamma' | \Gamma' \in C\}$$
$$C := mcss(\mathcal{FM}_{conv}, \Delta, H)$$
$$D := mcss(\mathcal{FM}_{adaptive}, \Delta', H)$$

This metric is defined as the ratio of the number of minimal critical sets of the conventional system that are a strict subset of a minimal critical set of the adaptive system to the total number of minimal critical sets of the conventional systems. A minimal critical set of the adaptive system that is not critical in the conventional system would be the result of introducing new critical failures together with the adaption mechanism and results in $d^{incl}_{qual} = -1$. It may be applied to a system whose conventional counterpart was already certified. If the adaptive systems has $d^{incl}_{qual} > 0$ then it is better than the conventional version and certification could be kept.

2) Quantitative Metrics: If all failures are statistically independent, then the result of the DCCA can be interpreted *quantitatively* using equation (10) for a single failure set and equation 11 that gives an upper bound for the probability of the hazard H to occur [5], [16].

$$P(\Gamma) = \prod_{\delta \in \Gamma} P(\delta) \tag{10}$$

$$P(H) \leq \sum_{\Gamma \in mcss(\mathcal{FM}, \Delta, H)} P(\Gamma) \tag{11}$$

Based on equation (10) we define the following quantitative metric analogous to the *qualitative minimal metric*.

Definition *5:* Quantitative minimal metric

The Quantitative minimal metric*for an adaptive system modeled by $\mathcal{FM}_{adaptive}$ with a corresponding non-adaptive system modeled by $\mathcal{FM}_{conv}$ for a set of failure modes Δ and a hazard H is defined as:*

$$d^{min}_{quan} := \frac{\max_{\Gamma \in mcss(\mathcal{FM}_{conv}, \Delta, H)} P(\Gamma)}{\max_{\Gamma' \in mcss(\mathcal{FM}_{adaptive}, \Delta', H)} P(\Gamma')} \tag{12}$$

The *Quantitative minimal metric* gives the ratio of probabilities of the minimal critical sets of the adaptive and the conventional system with the highest likelihood. Unfortunately it is not easy to define the real ratio of hazard probabilities of the adaptive and conventional system, as equation 11 is only an upper bound and no "sharp" probability measure. To obtain an exact quantitative measure all probabilities and the respective conditional probabilities are required. Nevertheless it is possible to compute and compare the probabilities that the hazard occurs in a given time interval based given minimal critical sets.

B. Results for Analysis of Case Study

We introduced transient failure automata as described in III-A into the model described in Sect. II-A. We considered every broken and thus unusable tool as failure. We also analyzed a second version with an additional failure, a possibly failing reconfiguration algorithm, i.e. wrong reconfiguration messages are sent due to network problems.

DCCA analyzes worst case behaviour, i.e. when a set of failures may lead to a hazard it is seen as critical. The hazard used in the DCCA of the model is the inability to process a workpiece in correct DIT order, i.e. a correct configuration of the production cell is no more possible. The hazard used for the DCCA of the model was the following predicate:

$$\neg(Control = None \wedge \neg ditFailure \wedge ditCapable)$$

This predicate is the negation of the fact that the system is not reconfigured at the moment ($Control = None$), there is no broken tool assigned to a robot ($ditFailure$) and the configuration is correct ($ditCapable$).

For the conventional model we computed nine disjoint sets of size one, containing each exactly one broken tool. As all failures appear as sets with one single element, no bigger minimal critical sets can be found due to monotonicity of *criticality*.

The results for the adaptive model showed a clear improvement according to our measurement. We computed six minimal critical sets of size three and 9 minimal critical sets of size four. The sets of size three consisted of either all broken tools of the same kind or of all broken tools of one robot. The sets of size four consisted of two pairs of broken tools for two robots, as the third one can only accomplish one task, no more production is possible, although all tools are available (but on the same robot). All bigger critical sets have critical subsets and are not minimal.

When a possibly failing reconfiguration was also implemented, modeling a failing network connection for example, there are an additional 9 minimal critical sets of size 2 (failing control and one failing tool).

The results from the safety analysis show that a clear increase in dependability according to all metrics. The *Qualitative minimal metric* is rather coarse giving $d^{min}_{qual} = 3$, or $d^{min}_{qual} = 2$ with a possible failing control mechanism. The weighted *Qualitative average metric* is more evened out, giving $d^{avg}_{qual} = 3.6$ or $d^{avg}_{qual} = 3$ respectively. This may be of course misleading, as there are minimal critical sets of size 2 despite $d^{avg}_{qual} = 3$. So, for this example the *Qualitative minimal metric* may be more appropriate as the system has failures with rather low occurrence probability in reality (the failing tools).

The *Qualitative inclusion metric* with $d^{incl}_{qual} = 1$ states that all minimal critical sets have been extended due to incorporating an adaption mechanism and there is no minimal critical set in the adaptive system that has no proper minimal critical subset in the conventional system. This means that from a safety point of view the adaptive system is clearly better and has no unforeseen failure due to the adaption mechanism.

If we assume a probability of a failing tool for one hour per week and a failing reconfiguration due to network errors for one hour every two weeks we get failure probabilities of roughly 0.006 and 0.003. Using this, the *Quantitative minimal metric* gives $d^{min}_{quan} = 27777.78$ and $d^{min}_{quan} = 336.00$ respectively.

V. Related Work

For verification and validation of adaptive systems, model checking is used by several other groups [8], [17]. Especially the Cadence SMV model checker is very attractive due to its high performance and usage of CTL temporal logic [8]. Quantitative analysis is possible using probabilistic model checking which is based on probabilistic logic like PCTL [18]. These techniques can compute the probability whether a given temporal formula holds.

Verification and Validation together with safety analysis are seen as very important for acceptance of adaptive and autonomous systems in [19]. The focus is more directed to learning systems, either offline or online, and complex systems where the additional problem of compositional verification is mentioned.

Different aspects of evaluation of adaptive systems are given in [6]. We addressed some of them, namely failure avoidance (robustness) and adaptivity which are closely connected in our view. Other aspects are mentioned, e.g. quality of service, granularity and time needed to adapt which are very interesting further topics. Developing formal methods for evaluation of adaptive systems is explicitly mentioned in [6] as very important step.

A different way to evaluate dependability of safety critical systems is to use Generalized Stochastic Petri Nets (GSPN) and its variants [20]. These methods focus on the quantitative analysis of an - already existing - qualitative result (e.g. a fault tree). The results of DCCA can be used as input data and be converted to a GSPN [21] for further probabilistic dependability analysis using Markov chains or Bayesian networks [20], [22]. However directly modeling adaptive systems as petri nets seems less practical for larger systems.

VI. Conclusion

Standard failure analysis like FMEA, FTA and DCCA are not directly applicable to adaptive systems as reconfiguration resolves many temporary hazards and restores functionality, separating system runs in functioning and reconfiguration phases. The DCCA approach was extended to address this problem of safety analysis for adaptive systems. The extended DCCA completeness theorem was proven and the CTL proof obligation for analyzing adaptive systems given.

From this several different qualitative and one quantitative metric for measuring dependability of adaptive systems were derived. Using these it is possible to guarantee (or falsify) an increase in dependability of an adaptive system in comparison to a conventional one.

The safety analysis with Adaptive-DCCA was exemplified with a case study of an adaptive production cell with reconfigurable robots and the qualitative metrics were applied to this case study. The result of this showed a clear increase of dependability by using a reconfiguration mechanism triggered by failing tools. Although qualitative metrics can be rather coarse in contrast to quantitative ones, they can be a big aid in early phases of development when the real probabilities of failures are unknown.

The next step is to formulate DCCA for different modeling tools not based on CTL temporal logic. An example is the LUSTRE [23] language which is integrated in the SCADE modeling tool. Together with a modeling technique for adaptive systems, this would integrate safety analysis for adaptive systems in a well accepted tool. This may augment the acceptance of adaptive systems in safety critical applications.

References

[1] C. Müller-Schloer, "Organic computing: on the feasibility of controlled emergence," in *CODES+ISSS '04*. NY, USA: ACM Press, 2004.

[2] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[3] R. E. McDermott, R. J. Mikulak, and M. R. Beauregard, *The Basics of FMEA*. Quality Resources, 1996.

[4] D. W. Vesley, D. J. Dugan, J. Fragole, J. M. II, and J. Railsback, *Fault Tree Handbook with Aerospace Applications*, NASA Office of Safety and Mission Assurance, NASA Headquarters, Washington DC 20546, August 2002.

[5] F. Ortmeier, W. Reif, and G. Schellhorn, "Deductive cause-consequence analysis (DCCA)," in *Proceedings of IFAC World Congress*, 2005.

[6] J. A. McCann and M. C. Huebscher, "Evaluation issues in autonomic computing." in *GCC Workshops*, 2004, pp. 597–608.

[7] K. L. McMillan, *Symbolic Model Checking*. Kluwer Academic Publishers, 1990.

[8] P. E. C. Pecheur, R. Simmons, "Formal verification of autonomy models: From livingstone to SMV," in *Agent Technology from a Formal Perspective*. Springer Verlag, 2006, pp. 103–113.

[9] M. Güdemann, F. Ortmeier, and W. Reif, "Formal modeling and verification of systems with self-x properties," in *Autonomic and Trusted Computing 2006, Proceedings*. Springer LNCS, 2006.

[10] D. A. P. E. M. Clarke Jr., O. Grumberg, *Model Checking*. The MIT Press, 1999.

[11] E. A. Emerson, "Temporal and modal logic," in *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1990, pp. 996–1072.

[12] F. Ortmeier and W. Reif, "Failure-sensitive specification: A formal method for finding failure modes," Institut für Informatik, Universität Augsburg, Tech. Rep. 3, 2004.

[13] T. A. Kletz, "Hazop and HAZAN notes on the identification and assessment of hazards," Inst. of Chemical Engineers, Rugby, England, Tech. Rep., 1986.

[14] A. Thums, "Formale fehlerbaumanalyse," Ph.D. dissertation, Universität Augsburg, Augsburg, Germany, 2004, (in German).

[15] F. Ortmeier, A. Thums, G. Schellhorn, and W.Reif, "Combining formal methods and safety analysis – the ForMoSA approach," in *Integration of Software Specification Techniques for Applications in Engineering*. Springer LNCS 3147, 2004.

[16] F. Ortmeier, "Formale sicherheitsanalyse," Ph.D. dissertation, Universität Augsburg, 2005, (in German).

[17] D. F. Gordon, "APT agents: Agents that are adaptive, predictable and timely," in *Proceedings of the First Goddard Workshop on Formal Approaches to Agent-Based Systems (FAABS'00)*, 2000.

[18] M. Kwiatkowska, G. Norman, and D. Parker, "Quantitative analysis with the probabilistic model checker PRISM," *Electronic Notes in Theoretical Computer Science*, vol. 153, no. 2, pp. 5–31, 2005.

[19] C. Pecheur, W. Visser, and R. Simmons, "Riacs workshop on the verification and validation of autonomous and adaptive systems," in *Conference report, AI Magazine, Fall 2001, Also available as NASA/TM-2001-210927*, 2001.

[20] S. Bernardi and S. Donatelli, "Stochastic petri nets and inheritance for dependability modelling." in *PRDC*, 2004, pp. 363–372.

[21] M. Malhotra and K. S. Trivedi, "Dependability modeling using petri-net based models," *IEEE Transactions on Reliability, , Vol. 44, No. 3*, pp. pp. 428–440, 1995.

[22] S. Bologna, E. Ciancamerla, M. Minichino, A. Bobbio, G. Franceschinis, L. Portinale, and R. Gaeta, "Comparison of methodologies for the safety and dependability assessment of an industrial programmable logic controller," *ESREL2001*, 2001.

[23] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous data-flow programming language LUSTRE," *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1305–1320, September 1991.