A Conceptual Framework for Safe Reconfiguration in Open System of Systems

Nikita Bhardwaj
Dept. of Software Engineering: Dependability
TU Kaiserslautern
Kaiserslautern, Germany
bhardwaj@cs.uni-kl.de

Peter Liggesmeyer
Dept. of Software Engineering: Dependability
TU Kaiserslautern
Kaiserslautern, Germany
liggesmeyer@cs.uni-kl.de

ABSTRACT

Software-intensive systems in application domains like autonomous agriculture, avionics and healthcare need to be more and more adaptive. Dynamic integration of components and a subsequent reconfiguration at runtime results in changed functional and non-functional properties of the system. Taking into consideration that these systems often operate in safety-critical environment, reconfiguration apropos of safety becomes a crucial concern. To this end, we introduce a conceptual framework that aids in safe reconfiguration of an open system-of-system, by the means of systematic monitoring and evaluation of the system and its components at runtime. We employ an example from autonomous agricultural domain to illustrate our concept and conclude this paper with discussing our planned future work.

CCS CONCEPTS

• **Computer systems organization** → *Embedded systems*; *Reliability*; *Redundancy*;

KEYWORDS

open systems, adaptivity, system of systems, safe reconfiguration, runtime safety, runtime monitoring

ACM Reference Format:
Nikita Bhardwaj and Peter Liggesmeyer. 2018. A Conceptual Framework for Safe Reconfiguration in Open System of Systems. In *SESoS'18: SESoS'18:IEEE/ACM 6th International Workshop on Software Engineering for Systems-of-Systems, May 29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3194754.3194759

1 INTRODUCTION

Open systems are capable of dynamic integration of components, devices or other systems at runtime to provide an emergent and collaborative functionality as a whole [1]. For such systems, adaptivity apropos of dynamic integration is a consequential requirement. An open and adaptive system is thus a system of systems capable of altering its own behaviour or architecture in response to either dynamic changes in the operational environment or the system itself.

Since these systems are often used in safety-critical domains like: autonomous agriculture, avionics and medical care, safety becomes an imperative non-functional requirement to fulfil. This brings two fundamental requirements for system of systems to be fulfilled: (1) The system must be capable of reconfiguration i.e. be able to adapt to dynamic changes regarding newly integrated components at runtime, and, (2) System reconfiguration must adhere to safety as an essential non-functional requirement, at all times in every operational situation.

As defined [2], safety is "the absence of catastrophic consequences on the user(s) and the environment". For open adaptive systems safety assurance is a challenging task. Random errors in system like wear&tear of physical components or unexpected environmental conditions might result in hazardous situations at runtime, thereby influencing system safety. As this information about system as well as its environment is unknown at design time, traditional safety analysis and assurance techniques are though still necessary, but not sufficient to guarantee safety at runtime. Hence, as proposed by Schneider et al. in [12], shifting certain elements of safety assurance from design time to runtime is necessary as well as beneficial, as it allows the system to make dynamic and adaptive decisions with respect to safety during execution. Besides, adaptive systems result in situations that demand use of runtime monitoring techniques. Depending upon the sources being used to monitor, runtime monitoring of safety properties shall provide a categorical assurance that the system is operating safe [9]. Moreover, safety monitoring acts as a fault tolerance technique, where the system is administered for violations in its behaviour and is brought back to the safe state in case of safety-critical deviations [3].

To this end, we introduce a conceptual framework that aids in safe reconfiguration of an open adaptive system at runtime. The concept behind the framework is to dynamically ensure safety by rigorous monitoring of operational status of system, its components and the environmental conditions during its execution, thereby managing system reconfiguration by evaluating system status in accordance with safety property. Owing to reconfiguration management at runtime, the framework is henceforth referred to as RM@RT.

The rest of this paper is structured as follows: In section 2, we present our conceptual framework with the help of a metamodel. We illustrate the fundamental elements of framework and demonstrate them with a relevant example from autonomous agriculture domain. We then conclude the paper with a brief discussion about

SESoS'18, May 29, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5747-0/18/05…$15.00
https://doi.org/10.1145/3194754.3194759

the planned and potential future research possibilities of our framework in section 3.

2 THE RM@RT FRAMEWORK

Figure 1 manifests the metamodel of our conceptual framework and its elements. The metamodel consists of an *Open Adaptive System* (OAS) as top element. We consider that the systems composing an OAS are also capable to reconfigure, and thus refer to them as *Adaptive Components* (ADC). This makes an OAS a composition of multiple adaptive components. *System Service Monitor* (SSM) is the fundamental element of the metamodel and is responsible for continuous monitoring of adaptive component apropos of deviations (explained later). Based on the deviations monitored, SSM triggers *Configuration Evaluation* (CE) to evaluate all potential configurations of the ADC with respect to safety property. This evaluation is achieved by means of *Knowledge Base* (KB), which acts as an information repository containing information essential for reconfiguration at runtime.

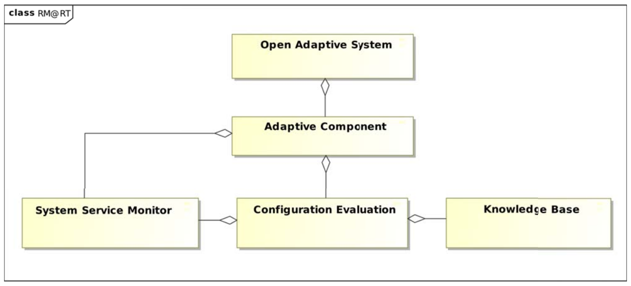

Figure 1: Metamodel for RM@RT Framework

2.1 Open Adaptive System

As illustrated in Figure 2, an OAS is composed of multiple adaptive components, where each of these components render a specific set of services. Together with all ADCs, the system provides an emergent and collaborative functionality. The system as well as each ADC is pre-engineered [11] i.e. all potential configurations it can achieve at runtime are pre-defined at design time. Its adaptation is thus restricted to these pre-defined configurations that suits best concerning the current operational environment. The configurations of an ADC differ from one another in terms of the services[1] they provide and the components they collaborate with (in order to provide those services) [1]. Any faulty or unplanned behaviour of collaborating component(s) or their rendered services at runtime influences the behaviour and certainly safety of the configuration itself.

This influences the safe behaviour of the ADC, as an ADC is as safe as its currently activated configuration. Furthermore, since the system is a composition of these adaptive components, unsafe behaviour of any component affects the safety of the entire system. This demonstrates a categorical dependency between safety of an OAS and safe reconfiguration of an ADC. In other words, selecting

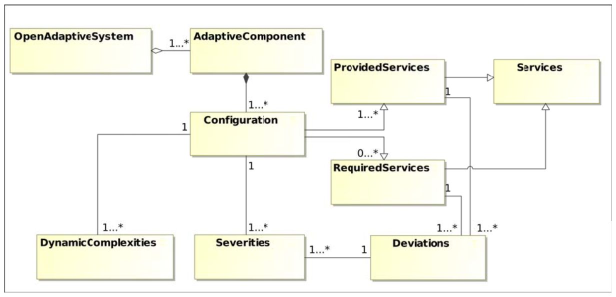

Figure 2: Associations between Open Adaptive System Entities

and activating a configuration based on its influence on safety at runtime is an imperative requisite to ensure safety for the OAS as a whole.

2.2 System Service Monitor

This element employs runtime monitoring for safe operation of an ADC by monitoring its potential configurations with respect to safety-critical deviations in its services. The behaviour of SSM is a declarative collection of safety rules, which is based on design time *hazard and severity analysis*, where each rule represents the planned behaviour of the service and its corresponding potential safety-critical deviations [1].

A deviation is an unplanned behaviour due to the discrepancy between the expected and the generated service either due to operational status of the system components or the environmental conditions. We classify these deviations based on their discrepancies attributed to time, value or provision. For instance, in case of an autonomous vehicle, if the service *VehicleSpeed (VehSp)* must be generated within 10ms from request time, but is generated after a delay of 2ms, then the service suffers a deviation attributed to time. For the autonomous vehicle, even such small time-deviation might result in a hazardous situation as the vehicle is unaware of its own speed for this duration. Moreover, if the speed generated after the delay (i.e. at 12ms) is incorrect, the safety of the component gets more vulnerable.

To refrain from such hazardous situations at runtime, the primary task of SSM is to monitor these deviations as they occur and trigger CE that evaluates the currently activated configuration and perform reconfiguration if necessary and possible. The possibility of reconfiguration depends upon the type and amount of configurations an ADC comprises. In case of redundant configurations, configurations that aid graceful-degradation or configurations based on levels-of-autonomy, reconfiguring to an another configuration due to deviations in the current one is possible. Otherwise, the system might need to shut-down and wait until the system is back in safe state.

The SSM can exhibit two different kinds of monitoring procedures: Time-triggered or event-triggered. In case time-triggered monitoring, the SSM periodically monitors the services for deviations and stays passive, for the rest of time, until the start of next interval. The disadvantage of such monitoring scheme is that any deviation occurring between two consecutive time periods is left

[1]A *service* can be defined as a behaviour that can be provided by any adaptive component to be used by any other adaptive component [11]

undetected. The event based monitoring scheme is a solution to avoid this, as in case of event-triggered monitoring, any deviation, at any point in time, would trigger monitor to detect it. However, this would result in an excessive overhead of monitoring inhibiting the component from executing its basic tasks [7] [4]. We are, therefore, currently working on finding a monitoring scheme that simultaneously allows safe operation and efficient performance of an ADC.

2.3 Configuration Evaluation

As the system and its ADCs are pre-engineered, all potential configurations that the system or its components can achieve are predefined at design time. Once SSM has detected a deviation in a service(s), it triggers CE with a reconfiguration request. Each ADC is equipped with an individual CE that performs reconfiguration by evaluating the currently activated configuration and all other potential configurations that this ADC can possibly be reconfigured to. The main objective of this evaluation is to assess these configurations in accordance with safety. This is done by carrying out a runtime risk analysis based on the information stored in KB. The risk analysis determines the risk of each configuration based on its dynamic complexity and the severity associated with its deviations.

The dynamic complexity is a measure of complexity of the currently executing configuration k (C_{F_k}) in a collaboration scenario i (CS_i), whereas the severity is the consequence of the hazardous situations resulted from the safety-critical deviations in its services [1]. The dynamic complexity (DC) is determined as:

$$DC_{C_{F_k}}^{CS_i} = CC_{C_{F_k}}^{CS_i} \times \left(II_{C_{F_k}}^{CS_i} \times OI_{C_{F_k}}^{CS_i}\right)^2 \tag{1}$$

where, CC is McCabe Cyclomatic Complexity [8] of C_{F_k} derived from control flow graphs. The latter half of equation (1) represents Information Flow Complexity [10], where $II_{C_{F_k}}^{CS_i}$ are the incoming interactions of C_{F_k} i.e. the components that C_{F_k} depends upon for its required services and $OI_{C_{F_k}}^{CS_i}$ are the number of components that depend upon C_{F_k} for its rendered services. To determine the severity associated with service deviations, we employ Hazard and Severity Analysis (HASA) at design time [1]. All potential deviations for each service are evaluated with respect to their causes and corresponding severities. Along with the dynamic complexity, the risk associated with a C_{F_k} in a scenario CS_i at runtime is calculated as:

$$RF_{C_{F_k}}^{CS_i} = NDC_{C_{F_k}}^{CS_i} \times SVT_{WCSv \in C_{F_k}}^{CS_i} \tag{2}$$

where, $NDC_{C_{F_k}}^{CS_i}$ is the normalized dynamic complexity and $SVT_{WCSv \in C_{F_k}}^{CS_i}$ is the worst-case severity of all the consequences due to service deviations.

Subsequent to the risk analysis, CE assigns a dynamic rank from best to worst to all potential configurations based on the increasing order of their risk. The configuration ranked best, the one with lowest associated risk, is activated after reconfiguration. The runtime risk analysis and assignment of dynamic ranks is not within the scope of this paper and is, thus not discussed further.

Evaluating the configurations for reconfiguration due to deviation(s) is not the only scenario when CE is triggered. We categorize the reconfiguration (RCf) trigger to CE into two categories: (1) Request-Driven and (2) Deviation-Driven. The request-driven reconfiguration is achieved either when a new component is integrated to the system, or when the execution mode of the system is requested to be altered e.g. when an autonomous vehicle is switched from autonomous mode to manual or vice-versa. Therefore, this trigger acts like a subsequent request to CE to carry out safe reconfiguration as a response to the structural or behavioural changes that have occurred in the system. The deviation-driven reconfiguration, as aforementioned, is an ensuing trigger for safe reconfiguration due to the safety-critical deviations monitored by SSM at runtime.

2.4 Knowledge Base

Knowledge Base (KB) is a repository that assists CE with reconfiguration at runtime. For each ADC, KB contains configuration specific information like: Total number of configurations, their set of required and rendered services, which configurations from other components these configurations collaborate with and the like. Additionally, it encompasses information about the dynamic complexities and the severities associated with each of these configurations.

The information maintained in KB is acquired from system analyses at design time. For instance, severities associated with the deviations in services is obtained from Hazard and Severity Analysis. A service can suffer multiple deviations and thus can result in different hazardous situations with assorted severities. For each service, all potential deviations along with their associated severities is obtained from hazard and severity analysis and is stored in KB. At runtime, when CE is triggered for evaluation, it accesses information about the dynamic complexity and severities of the configurations to be evaluated from KB.

Case Study: Figure 3 demonstrates an example from agricultural automotive domain. There is a *Tractor* that is integrated with a *Baler* implement in order to perform baling on the field. The tractor consists of a *SwathScanner* that aids *Baler* with services required in carrying out baling. Both *Baler* and *SwathScanner* are adaptive components with multiple configurations. The configurations labelled as *BasicCf* are for the manual mode of driving and can be activated only when the driver is present. The others marked as *AutoCf* are for autonomous baling operation, with a certain degree of autonomy. To render *Baling* service, *Baler* requires information about the volume, flow and the location of the swath, which is provided as *SwVolFlow* and *SwLoc* services by *SwathScanner.* To do so, scanner further requires information about *VehSp* generated by the components *SpeedSensor* (SpSr). The two *AutoCf* configurations of the scanner are redundant and differ merely from the sensors generating the *VehSp* service.

We consider a scenario where the system is performing autonomous baling and thus both *Baler* and *SwathScanner* have *AutoCf1* configurations activated. During baling, *SpSr1* suffers a time-deviation because of wear&tear and doesn't generate *VehSp* within the permissible duration. SSM monitors this deviation and triggers

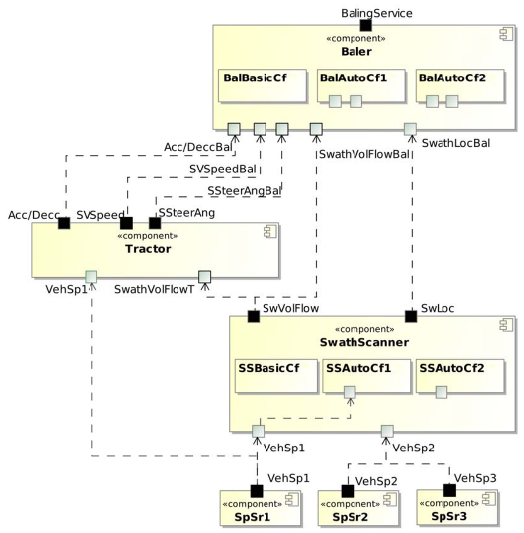

Figure 3: Configurations and Services of Baler and SwathScanner

CE with an evaluation request. Based on the information extracted from the KB, CE performs risk assessment for both *AutoCf1* and *AutoCf2* (as the potential configuration). Since the speed sensors SpSr2 and SpSr3 do not suffer any deviation, the risk associated with *AutoCf1* is certainly higher than with *AutoCf2*. As a result, CE assigns *AutoCf2* with a higher rank and reconfigures *SwathScanner*, thereby activating *AutoCf2*.

The aforementioned reconfiguration brings system both to a safe and reliable state since: On the one hand, the risk associated with continuing system operation with *AutoCf1* is evidently high as, not knowing the vehicle speed in an autonomous mode brings the entire system to a hazardous state where it is unaware of its whereabouts. Moreover, there is a likelihood that the value generated after the delay *VehSp* might be influenced by the wear&tear too and hence, be incorrect. On the other hand, the component *SwathScanner* has a potential configuration that suffer no safety-critical deviations and hence can take over and continue rendering the expected functionality, like before.

3 CONCLUSION AND FUTURE WORK

In this paper we propose, via a metamodel, a conceptual framework responsible for safe reconfiguration of an open adaptive system at runtime. The core idea is to exploit runtime safety monitoring where, in the event of any safety-critical deviation, a reconfiguration request shall be triggered for an Adaptive Component (ADC) that enables it to safely migrate from currently activated configuration to a new configuration. One of the many advantages of the presented approach is that it is domain independent and is thence, applicable to open system of systems from domains other than automotive, e.g. medical care. Runtime safety monitoring aids system to be monitored, analysed and evaluated in accordance with safety property, and in case of any safety-critical anomalies with respect to the planned behaviour, system is reconfigured to a safe state.

We see much research scope with respect to incorporating models at runtime [6] [13] into our framework. As an open adaptive system is a collection of system of systems, realization of the framework incorporating runtime models would require the models to be capable of composability and thus be modular. Although challenging, yet we believe this is achievable as our framework is modular.

Nevertheless, the introduced approach is work in progress, and therefore necessitates imperative refinements. We are currently implementing the proposed framework using case studies from autonomous agriculture to evaluate it and ascertain associated challenges, if any. Besides, we aim to employ runtime models into our framework and create adaptation logic to assist safety based dynamic adaptation [5] at runtime. To our best knowledge, reflection models, causal connection models and adaptation models are quite a few runtime models that shall play a fundamental role in our work.

REFERENCES

[1] Nikita Bhardwaj and Peter Liggesmeyer. 2017. A Runtime Risk Assessment Concept for Safe Reconfiguration in Open Adaptive Systems. *LNCS, Springer, Cham* 10489, 309–316. https://doi.org/10.1007/978-3-319-66284-8_26

[2] Algirdas Avizienis et al. 2004. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Trans. Dependable Secur. Comput.* (Jan. 2004), 11–33. https://doi.org/10.1109/TDSC.2004.2

[3] Amina Mekki-Mokhtar et al. 2012. Safety Trigger Conditions for Critical Autonomous Systems. In *The 18th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2012)*. https://doi.org/10.1109/PRDC.2012.22

[4] Borzoo Bonakdarpour et al. 2012. Time-Triggered Program Self-Monitoring. 260–269. https://doi.org/10.1109/RTCSA.2012.16

[5] Brice Morin et al. 2009. Models@ Run.time to Support Dynamic Adaptation. (2009), 44–51. https://doi.org/10.1109/MC.2009.327

[6] Holger Giese et al. 2014. *Living with Uncertainty in the Age of Runtime Models*. Springer International Publishing, 47–100. https://doi.org/10.1007/978-3-319-08915-7_3

[7] Mathilde Machin et al. 2014. Specifying Safety Monitors for Autonomous Systems using Model-checking. 262–277. https://doi.org/10.1007/978-3-319-10506-2_18

[8] Thomas J. McCabe. 1976. A Complexity Measure. *IEEE Transactions on Software Engineering, Vol.:SE-2, Issue:4* (Dec. 1976), 208–220. https://doi.org/10.1109/TSE.1976.233837

[9] John Rushby. 2008. Runtime Certification. Springer-Verlag, 21–35. https://doi.org/10.1007/978-3-540-89247-2_2

[10] Henry Sallie and Dennis Kafura. 1981. Software Structure Metrics Based on Information Flow. *IEEE Transactions on Software Engineering, Vol.:SE-7, Issue:5* (Sept. 1981), 510 – 518. https://doi.org/10.1109/TSE.1981.231113

[11] Daniel Schneider. 2014. Conditional Safety Certification for Open Adaptive Systems. *TU Kaiserslautern* (2014), 1–205.

[12] Daniel Schneider and Mario Trapp. 2010. Conditional Safety Certificates in Open Systems. *Proceedings of the 1st Workshop on Critical Automotive Applications: Robustness &#38; Safety*, 57–60. https://doi.org/10.1145/1772643.1772660

[13] Sebastian Waetzoldt and Holger Giese. 2014. Classifying Distributed Self-* Systems Based on Runtime Models and Their Coupling. In *Proceedings of the 9th Workshop on Models@run.time co-located with 17th International Conference on Model Driven Engineering Languages and Systems*. CEUR-WS, 11–20.