Scientific Research Publishing

Journal of Computer and Communications, 2018, 6, 34-48
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

# Representation of Categorical Specification of Self-Configurations in Reactive Autonomic Systems Framework

**Ming Zhu[1], Heng Kuang[2], Jing Li[1]**

[1]College of Computer Science and Technology, Shandong University of Technology, Zibo, China
[2]Huawei Canada, Markham, Canada
Email: zhu_ming@sdut.edu.cn, heng.kuang@huawei.com, li_jing@sdut.edu.cn

**How to cite this paper:** Zhu, M., Kuang, H. and Li, J. (2018) Representation of Categorical Specification of Self-Configurations in Reactive Autonomic Systems Framework. *Journal of Computer and Communications*, **6**, 34-48.
https://doi.org/10.4236/jcc.2018.612003

**Received:** November 27, 2018
**Accepted:** December 17, 2018
**Published:** December 20, 2018



## Abstract

Software complexity crisis brings huge obstacle to further progress in IT industry. To alleviate this problem, researchers are asked to build systems which can benefit from automation. With autonomic behavior, the real-time reactive systems can be more self-managed and adaptive to their environment. However, most of current formal approaches fail to specify such kind of system. In this paper, we proposed an approach to formally specify reactive autonomic systems. First, we used category theory to formalize reactive autonomic systems; then we focused on the categorization of self-configurations and work flows of reactive autonomic systems, and finally we used XML to specify the categorical models. In doing so, it can help to build the foundation of reactive autonomic systems with autonomic features and verify emergent behaviors.

## Keywords

Reactive Autonomic System, Category Theory, XML, Self-Configuration

## 1. Introduction

Real-time reactive systems can be very complex, and difficult to test and error-finding. Race conditions in real-time reactive systems are hard to be found by only inputting sample data and checking the results, as certain errors are time-based and only occur when processes send or receive data at particular time, in particular in sequence or after learning. In order to find those errors, all possible state combinations of the processes have to be executed, which are exponential in the number of states [1]. Formal method could provide systems

with known safety properties since a formal specification can be used to check for particular types of errors and as inputs for model checking. Category theory has been proposed as a framework to offer specification structure. It has a rich body of theory to reason objects and their relations. Moreover, category theory adopts a correct by construction approach by which components can be specified, proved and composed in the way of preserving their properties. Unfortunately, most of current formal approaches fail to specify Reactive Autonomic System (RAS) and do not address well on verifying emergent behaviors, which is an important characteristic for the RAS.

To handle above mentioned problem, as a continuation of research [2] [3] [4], we first discuss how to model RAS and its configurations in category theory; then, we show how to transform the self-configuration properties to the categorical representations, and we use XML to specify the representations. The rest of this paper is organized as follows: Section 2 gives an overview of the related work. Section 3 introduces background knowledge required to understand the remaining content of the paper. Section 4 introduces categories that represent RAS model and its configurations respectively. Section 5 shows how to transform the categorical self-configuration and the properties into categories, and uses XML to specify the categories. Finally, Section 6 concludes the work.

## 2. Related Work

In this section, research work related to this paper is introduced.

### 2.1. Real-Time Reactive Systems

Seshia proposes an architecture for automatically recovering a class of reactive systems from run-time failures [5]. The system comprises executions which can be divided into several rounds and each round performs a new unit of work. The framework leverages parallelism to proactively explore the space of repairs before a failure is occurred. Paper [6] presents a self-adapting loop according to system-specific adaptation knowledge that includes the types and properties of autonomic components, behavior constraints as well as strategies for adaptation. This system is an integral part of a real-time system which controls the behavior of computing environment and evaluates its global behavior.

### 2.2. Formal Methods

For modeling concurrency, category theory is used to model, analyze, and compare Transition System, Trace Language, Event Structure, Petri nets, and other classical models of concurrency [7] [8] [9]. Mackworth and Zhang describe a Constraint-Based Agent (CBA) design approach which includes two formal models: Constraint Nets and Timed ∀-automata [10]. A constraint net can model agents and their environment symmetrically as dynamical systems; timed ∀-automata can specify desired real-time dynamic behaviors of those situated agents. Paper [11] introduces a formal language model which formalizes agent-

environment interaction in a multi-agent framework called Conversational Grammar Systems (CGS). This system provides a model with a high degree of flexibility. Based on eco-grammar systems, the formal model used in this paper can be defined as an evolutionary multi-agent system. Category theory is applied to study relationships between geometrical models for concurrency and classical models [12].

## 3. Background

In this section, background and work related to our research are introduced.

### 3.1. Reactive Autonomic Systems (RAS)

The Reactive Autonomic Systems (RAS) architecture model (**Figure 1**) is a four-layer architecture that consists of Reactive Autonomic Objects (RAO), Reactive Autonomic Components (RAC), Reactive Autonomic Component Groups (RACG) as well as the RAS. The autonomic features are implemented by RAO Leaders (RAOL), RAC Supervisors (RACS) and RACG Managers (RACGM) at the RAC, RACG as well as RAS layer respectively [2]. In this layered architecture model, each tier communicates only with the tier immediately above or below it. Thus, the independence of those tiers makes their modularity, encapsulation, hierarchical decomposition and reuse possible.

1) RAO is modeled as a labeled transition system augmented with ports, resources, attributes and the logical assertion on those attributes as well as time constraints [13]. More specifically, it is modeled as a 9-tuple ($P$, $\varepsilon$, $\Theta$, $X$, $L$, $\Phi$, $\Lambda$, $\gamma$, $R$) where $P$, $\varepsilon$, $\Theta$, $X$, $L$, $\Phi$, $\Lambda$, $\gamma$ are specified as in [14]:

- $P$ is a finite set of ports associated with each port-type and the null-type $P_0$ whose only port is the null port $P_0$.
- $\varepsilon$ is a finite set of events and includes the silent-event tick.
- $\Theta$ is a finite set of states where $\Theta_0$: $\Theta$, is the initial state; there is no final state.
- $X$ is a finite set of typed attributes: abstract data types and port reference types.
- $L$ is a finite set of LSL traits for the abstract data type used in $X$.

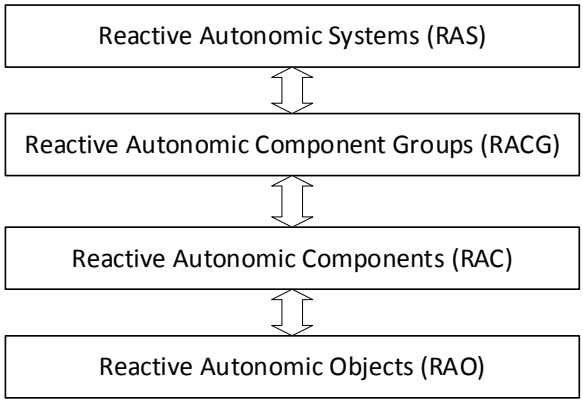


**Figure 1.** RASF architecture model.

- $\Phi$ is a function-vector ($\Phi_s$, $\Phi_{at}$) which $\Phi_s$ associates with each state $\Theta$ *a* set of sub states and $\Phi_{at}$ associates with each state $\Theta$ a set of attributes.
- $\Lambda$ is a finite set of transition specifications.
- $\gamma$ is a finite set of time-constraints.
- $R$ models the set of resources available locally for the object to support its functionality.

2) RAC is a set of synch.

Ronously communicating RAO, where one of the RAO is assigned as a leader (RAOL) of the rest (workers). The workers are responsible for reactive tasks, while the RAOL works on autonomic tasks such as coordinating the self-monitoring at component level. Thus, the RAOL has a different set of states from the workers, which states are autonomic behavior related instead of reactive behavior related. The reactive and autonomic natures of formal specifications for the RAOL enable them to implement autonomic functionalities in a real-time reactive system. In order to coordinate the work as well as communication between the RAO, a RAC specification consists of *Members*, *Configure*, *Leader*, *Supervisor*, *Neighbors and Repository*. The RAC is the minimum centralized Reactive Autonomic Element (RAE) that has the ability of self-management in RAS.

Similarly to the RAO, the reactive behavior of a RAC consists of *n* collaborating. RAO is specified as a 9-tuple ($P^{syn}$, $\varepsilon^{syn}$, $\Theta^{syn}$, $X^{syn}$, $L^{syn}$, $\Phi^{syn}$, $\Lambda^{syn}$, $\gamma^{syn}$, $R^{syn}$) [13]:

- $P^{syn}$ is a set of port-types allowing for a synchronous communication between the RAO.
- $\varepsilon^{syn}$ is a union of all $\varepsilon_i$ where $i$: [1 ... $n$].
- $\Theta^{syn}$ is a finite set of reachable and valid Synchronous Production Machine (SPM) state.
- $X^{syn}$ is a union of the finite sets $X_1^{syn}$, …, $X_n^{syn}$.
- $L^{syn}$ is a union of the finite sets of Larch Specification Language (LSL) traits for Abstract Data Type (ADT) used in the RAO.
- $\Phi^{syn}$ is a function-vector ($\Phi_s^{syn}$, $\Phi_{at}^{syn}$, $\Phi_\gamma^{syn}$) that $\Phi_s^{syn}$ associates with each SPM state $\Theta^{syn}$ *a* set of sub states and $\Phi_{at}^{syn}$ associates with each SPM state $\Theta^{syn}$ the uniton of the set of attributes a set of attributes $\Phi_{at1}(\Theta_1^{syn})$, …, $\Phi_{atn}(\Theta_n^{syn})$: $\Phi_\gamma^{syn}$ associates each SPM state $\Theta^{syn}$ with a subset of $R^{syn}$.
- $\Lambda^{syn}$ is a finite set of transition specifications.
- $\gamma^{syn}$ is a finite set of time-constraints.
- $R^{syn}$ is a set of resources available in the RAO; it is defined as a union of all $R_i$: $i$: [1 ... $n$].

3) RACG is a set of centralized or distributed RAC that cooperate in fulfillment of group tasks by synchronous communications. It is the minimum RAE which can independently accomplish complete real-time reactive tasks in RASF. The autonomic behavior at group level is coordinated by a supervisor (RACS).

4) RAS is made up of centralized or distributed RACG with asynchronous communication. It provides an integrated interface for users to delegate computing tasks, monitor systems and manage repositories. A manager (RACGM) is

responsible for coordinating autonomic behavior at system level.

### 3.2. Self-Configuration in Reactive Autonomic Systems

Self-configuration is an essence of RAS. Components in RAS are able to configure themselves automatically according to high level policies (business level objectives), which specify what is required instead of how they are implemented. The self-configuration work flows of RAC, RACG and RAS are represented in sequence diagrams are illustrated in **Figures 2-4** respectively.

### 3.3. Category Theory

Category theory has a rich body of theory to reason objects as well as their relations, and it is abstract enough for a wide range of different specification languages [15]. Categorytheory for the software specification has adopted a correct by construction approach by which components are specified, proved and composed in the way of preserving their properties [15]. Moreover, category theory can provide techniques to manipulate and reason diagrams for building hierarchies of system complexity, allowing systems to be used as components of more complex systems and inferring properties of the systems from their configurations [12].

As there is no such kind of formalization for self-configurations of RAS, we propose to use category theory to formalize the self-configurations. To understand this paper, we introduce the definition of category.

**Definition 1:** A category consists of objects and morphisms. $A$ morphism $f$: $A \rightarrow B$ has object $A$ as its domain and object $B$ as its codomain, respectively. If there are morphisms $f$: $A \rightarrow B$ and $g$: $B \rightarrow C$, then there is also a morphism $g f$: $A$

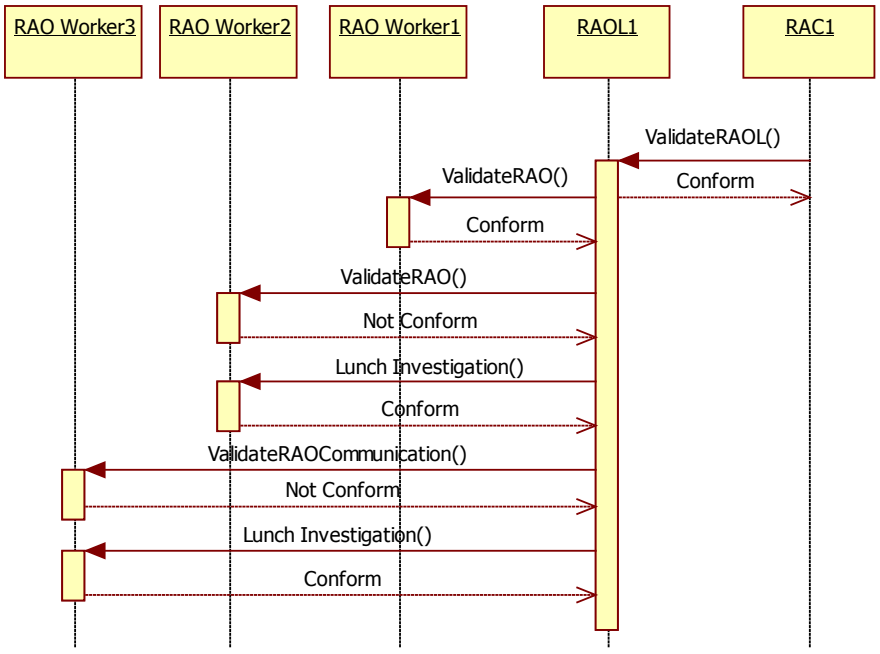


**Figure 2.** RAC self-configuration work flow.

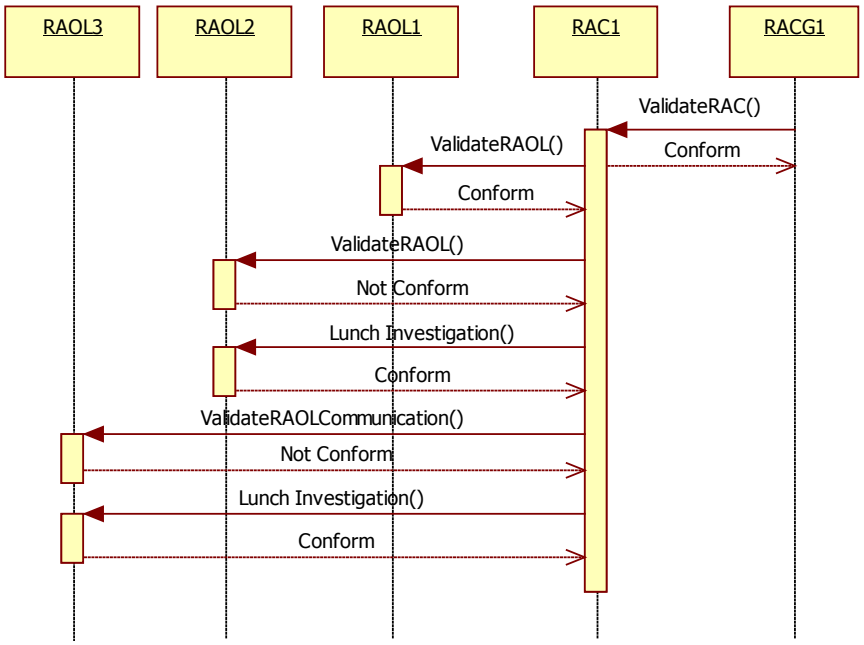


**Figure 3.** RACG self-configuration work flow.

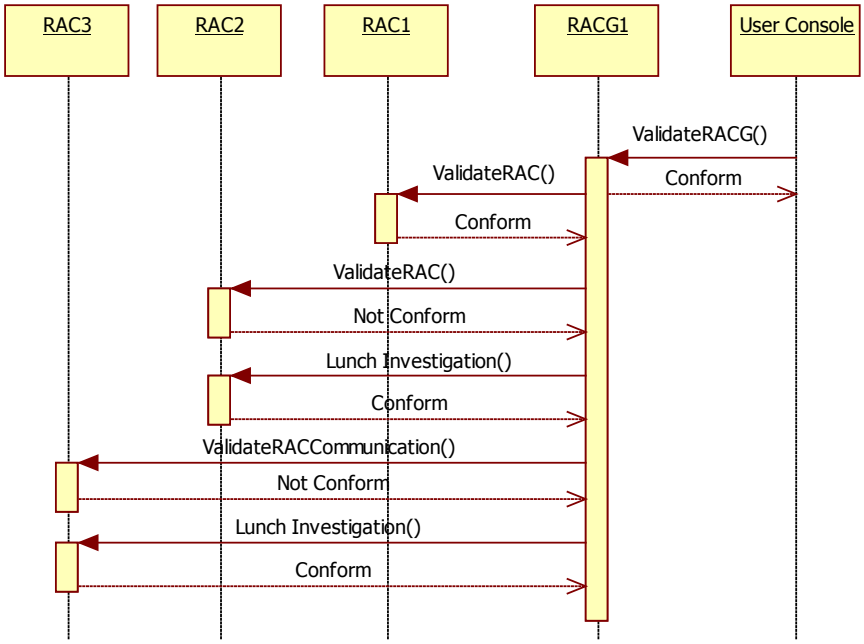


**Figure 4.** RAS self-configuration work flow.

$\rightarrow C$ called their composition. Composition is associative: $(h \circ g) \circ f = h \circ (g \circ f)$. Every object $X$ has an identity morphism $Id_X$. For every morphism $f: A \rightarrow B$, $Id_B \circ f = f = f \circ Id_A$.

## 4. Categorical Representations of RAS Model

In order to categorize the self-configuration of RAS, we need to formalize RAS model by using category theory first. In this section, we build categorical repre-

sentations of RAS model including RAC, RACG, and RAS.

**Proposition 1:** RAC can be specified as a category RAC with a set of objects |RAC| and morphisms so that for each $RAO_i$, $RAO_j \in$ |RAC|, there is a set of morphisms $f$: RAC ($RAO_i$, $RAO_j$) mapping the $RAO_i$ to $RAO_j$ which indicate the *communication* between them as $f$: i $RAO_i \to RAO_j$.

**Proof:** All what we need is to prove: 1) the existence of composition and identity morphism, and 2) prove associativity. Let $RAO_1$, $RAO_2$ and $RAO_3$ be three $RAO$ such that $RAO_1$ can interact with $RAO_2$, which can interact with $RAO_3$. Then $RAO_1$ can communicate with $RAO_3$ (indirectly through $RAO_2$), which means the existence of a composition of morphisms between $RAO_1$ and $RAO_3$. The identity morphism does exist as a natural representation of internal *communications*. Let $f$, $g$ and $h$ be the morphisms such that $f$: $RAO_1 \to RAO_2$, $g$: $RAO_2 \to RAO_3$ and $h$: $RAO_3 \to RAO_4$. It is clear that $h \circ (g \circ f) = (h \circ g) \circ f$.

**Proposition 2:** The RACG may also be specified as a category RACG with a set of objects |RACG| and morphisms such that for each $RAC_m$, $RAC_n \in$ |RACG|, there is a set of morphisms $f$: RACG ($RAC_m$, $RAC_n$) mapping the $RAC_m$ to the $RAC_n$ that indicate the *communication* between them as $f$: $RAC_m \to RAC_n$.

**Proof:** All what we need is to prove: 1) the existence of composition and identity morphism, and 2) prove associativity. Let $RAC_1$, $RAC_2$ and $RAC_3$ be three $RAC$ such that$RAC_1$ can interact with $RAC_2$, which can interact with $RAC_3$. Then $RAC_1$ can communicate with $RAC_3$ (indirectly through $RAC_2$), which means the existence of a composition of morphisms between $RAC_1$ and $RAC_3$. The identity morphism does exist as a natural representation of internal *communications*. Let $f$, $g$ and $h$ be the morphisms such that $f$: $RAC_1 \to RAC_2$, $g$: $RAC_2 \to RAC_3$ and $h$: $RAC_3 \to RAC_4$. It is clear that $h \circ (g \circ f) = (h \circ g) \circ f$.

**Proposition 3:** The RAS may also be specified as a category RAS with a set of objects |RAS| and morphisms such that for each $RACG_x$, $RACG_y \in$ |RAS|, there is a set of morphisms $f$: RAS ($RACG_x$, $RACG_y$) mapping the $RACG_x$ to the $RACG_y$ that indicate their *communications* as $f$: $RACG_x \to RACG_y$.

**Proof:** All what we need is to prove: 1) the existence of composition and identity morphism, and 2) prove associativity. Let $RACG_1$, $RACG_2$ and $RACG_3$ be three $RACG$ such that $RACG_1$ can communicate with $RACG_2$, which can interact with $RACG_3$. Then $RACG_1$ can interact with $RACG_3$ (indirectly through $RACG_2$), which means the existence of a composition of morphisms between $RACG_1$ and $RACG_3$. The identity morphism does exist as a natural representation of internal *communications*. Let $f$, $g$ and $h$ be the morphisms such that $f$: $RACG_1 \to RACG_2$, $g$: $RACG_2 \to RACG_3$ and $h$: $RACG_3 \to RACG_4$. It is clear that $h \circ (g \circ f) = (h \circ g) \circ f$.

Figure 5 illustrates the categorical representations and the corresponding proofs in Proposition 1 - 3.

## 5. Categorical Representations of Self-Configurations of RAS Model

Based on the categorical representations of RAS model in Section 3, in this section,

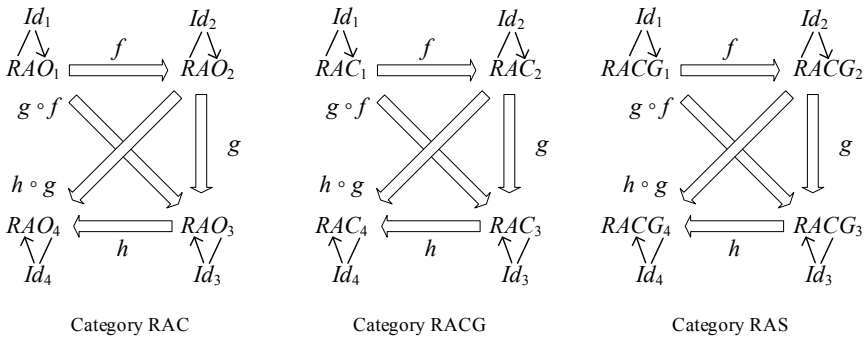


Figure 5. Categorical representations of RAS model.

we build categorical representations of self-configurations of RAS model, and use xml to specify the categories.

**Proposition 4:** The configuration of a RAC is a category denoted as CONFIG (RAC), where objects are *RAO* and morphisms are *connections* between those *RAO* as CONFIG ($RAO_i$, $RAO_j$) or CONFIG ($RAO_j$, $RAO_i$).

**Proof:** The proof of Proposition 4 is similar to the proof of Proposition 1, where the morphism *connection* in Proposition 4 is similar to the morphism *communication* in Proposition 1.

The XML specification of the category CONFIG (RAC) can be described as follows:

<CATEGORY name = "*RAC-Configuration*">
<OBJECT>
<OBJECT name = "*RAOi*" type = "*RAO-Typei*"/>
<OBJECT name = "*RAOj*" type = "*RAO-Typej*"/>
</OBJECT>
<MORPHISM>
<MORPHISM name = "*Connectionm*" type = "*Connection-Typem*"/>
<FROM-OBJECT name = "*RAOi*" type = "*RAO-Typei*"/>
<TO-OBJECT name = "*RAOj*" type = "*RAO-Typej*"/>
</MORPHISM>
</CATEGORY>

**Proposition 5:** The configuration of a RACG is the category denoted as CONFIG (RACG), where objects are *RAC* and morphisms are *connections* between those *RAC* as CONFIG ($RAC_i$, $RAC_j$) or CONFIG ($RAC_j$, $RAC_i$).

**Proof:** The proof of Proposition 5 is similar to the proof of Proposition 2, where the morphism *connection* in Proposition 5 is similar to the morphism *communication* in Proposition 2.

The XML specification of the category CONFIG (RACG) can be described as follows:

<CATEGORY name = "*RACG-Configuration*">
<OBJECT>
<OBJECT name = "*RACi*" type = "*RAC-Typei*"/>
<OBJECT name = "*RACj*" type = "*RAC-Typej*"/>

</OBJECT>
<MORPHISM>
<MORPHISM name = "*Connectionm*" type = "*Connection-Typem*"/>
<FROM-OBJECT name = "*RACi*" type = "*RAC-Typei*"/>
<TO-OBJECT name = "*RACj*" type = "*RAC-Typej*"/>
</MORPHISM>
</CATEGORY>

**Proposition 6:** The configuration of a RAS is a category denoted as CONFIG (RAS), where objects are *RACG* and morphisms are the *connections* between those *RACG* as CONFIG ($RACG_i$, $RACG_j$) or CONFIG ($RACG_j$, $RACG_i$).

**Proof:** The proof of Proposition 6 is similar to the proof of Proposition 3, where the morphism *connection* in Proposition 6 is similar to the morphism *communication* in Proposition 3.

The XML specification of the category CONFIG (RAS) can be described as follows:

<CATEGORY name = "*RAS-Configuration*">
<OBJECT>
<OBJECT name = "*RACGi*" type = "*RACG-Typei*"/>
<OBJECT name = "*RACGj*" type = "*RACG-Typej*"/>
</OBJECT>
<MORPHISM>
<MORPHISM name = "*Connectionm*" type = "*Connection-Typem*"/>
<FROM-OBJECT name = "*RACGi*" type = "*RACG-Typei*"/>
<TO-OBJECT name = "*RACGj*" type = "*RACG-Typej*"/>
<MORPHISM>
</CATEGORY>

As introduced in Section 1, the self-configurations of RAC, RACG, and RAS follows the work flow illustrated in Figures 2-4. For each work flow, we can build a category for it, and use XML to specify the corresponding category.

**Proposition 7:** The self-configuration work flow of RAC is a category denoted as CONWORKFLOW (RAC), where its objects are messages *Validate RAOL*, *ValicateRAO*, *LunchInvestigation*, *ValidateRAOCommunication*, *Conform*, *NotConform*, and the morphisms denote the relationship *before* between the occurrences of objects $before_1$: *ValidateRAOL* → *Conform*, $before_2$: *ValidateRAOL* → *NotConform*, $before_3$: *ValidateRAO* → *Conform*, $before_4$: *ValicateRAO* → *NotConform*, $before_3$: *ValidateRAOCommunication* → *Conform*, $before_6$: *ValicateRAOCommunication* → *NotConform*, $before_7$: *NotConform* → *LunchInvestigation*.

**Proof:** All what we need is to prove: 1) the existence of composition and identity morphism, and 2) prove associativity. Let $Obj_1$, $Obj_2$ and $Obj_3$ be three messages such that $Obj_1$ occurs before $Obj_2$, which occurs before $Obj_3$. Then $Obj_1$ occurs before $Obj_3$ (indirectly through $Obj_2$), which means the existence of a composition of morphisms between $Obj_1$ and $Obj_3$. The identity morphism does exist as a natural representation of interactions with occurrence. Let *f*, *g* and *h* be

the morphisms such that *f*: $Obj_1 \to Obj_2$, g: $Obj_2 \to Obj_2$ and h: $Obj_3 \to Obj_4$. It is clear that $h \circ (g \circ f) = (h \circ g) \circ f$.

The XML specification of the category CONWORKFLOW (RAC) can be described as follows:

<CATEGORY name = "Self-Configuration-Work-Flow-RAC">
<OBJECT>
<OBJECT name = "*ValidateRAOL*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*ValicateRAO*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*LaunchInvestigation*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*ValidateRAOCommunication*"
type = "*Work-Flow-Action*"/>
<OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
</OBJECT>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValidateRAOL*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValidateRAOL*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValicateRAO*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValicateRAO*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValidateRAOcommunication*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "Preorder"/>
<FROM-OBJECT name = "*ValidateRAOcommunication*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>

<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*LaunchInvestigation*"
type = "*Work-Flow-Action*"/>
</MORPHISM>
</CATEGORY>

**Proposition 8:** The self-configuration work flow of RACG is a category denoted as CONWORKFLOW (RACG), where its objects are messages *ValidateRAC*, *ValicateRAOL*, *LunchInvestigation*, *ValidateRAOLCommunication*, *Conform*, *NotConform*, and the morphisms denote the relationship *before* between the occurrences of objects $before_1$: *ValidateRAC* → *Conform*, $before_2$: *ValidateRAC* → *NotConform*, $before_3$: *ValicateRAOL* → *Conform*, $before_4$: *ValicateRAOL* → *NotConform*, $before_3$: *ValidateRAOLCommunication* → *Conform*, $before_6$: *ValidateRAOLCommunication* → *NotConform*, $before_7$: *NotConform* → *LunchInvestigation*.

**Proof:** The proof of Proposition 8 is omitted, as it is similar to the proof of Proposition 7.

The XML specification of the category CONWORKFLOW (RACG) can be described as follows:

<CATEGORY name = "Self-Configuration-Work-Flow-RACG">
<OBJECT>
<OBJECT name = "*ValidateRAC*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*ValicateRAOL*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*LaunchInvestigation*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*ValidateRAOLCommunication*"
type = "*Work-Flow-Action*"/>
<OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
</OBJECT>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValidateRAC*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValidateRAC*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValicateRAOL*"
type = "*Work-Flow-Action*"/>

<TO-OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValicateRAOL*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValidateRAOLCommunication*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "Preorder"/>
<FROM-OBJECT name = "*ValidateRAOLCommunication*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*LaunchInvestigation*"
type = "*Work-Flow-Action*"/>
</MORPHISM>
</CATEGORY>

**Proposition 9:** The self-configuration work flow of RAS is a category denoted as CONWORKFLOW(RAS), where its objects are messages *ValidateRACG*, *ValicateRAC*, *LunchInvestigation*, *ValidateRACCommunication*, *Conform*, *NotConform*, and the morphisms denote the relationship *before* between the occurrences of objects $before_1$: *ValidateRACG* → *Conform*, $before_2$: *ValidateRACG* → *NotConform*, $before_3$: *ValidateRAC* → *Conform*, $before_4$: *ValicateRAC* → *NotConform*, $before_5$: *ValidateRACCommunication* → *Conform*, $before_6$: *ValidateRACCommunication* → *NotConform*, $before_7$: *NotConform* → *LunchInvestigation*.

**Proof:** The proof of Proposition 9 is omitted, as it is similar to the proof of Proposition 7.

The XML specification of the category CONWORKFLOW (RACG) can be described as follows:

<CATEGORY name = "Self-Configuration-Work-Flow-RACG">
<OBJECT>
<OBJECT name = "*ValidateRACG*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*ValicateRAC*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*LaunchInvestigation*" type = "*Work-Flow-Action*"/>
<OBJECT name = "*ValidateRACCommunication*"
type = "*Work-Flow-Action*"/>
<OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>

<OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
</OBJECT>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValidateRACG*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValidateRACG*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValicateRAC*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValicateRAC*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*ValidateRACCommunication*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*Conform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "Preorder"/>
<FROM-OBJECT name = "*ValidateRACCommunication*"
type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<MORPHISM>
<MORPHISM name = "*Before*" type = "*Preorder*"/>
<FROM-OBJECT name = "*NotConform*" type = "*Work-Flow-Action*"/>
<TO-OBJECT name = "*LaunchInvestigation*"
type = "*Work-Flow-Action*"/>
</MORPHISM>
</CATEGORY>

## 6. Conclusion

Due to the difficulties of test and error-finding in real-time reactive systems, in this paper, we proposed an categorical approach to formally specify reactive au-

tonomic systems and the self-configuration work flows, which can help to check for particular types of errors and as inputs for model checking. Moreover, we used XML to specify the constructed categorical models, which can help to build the foundation of reactive autonomic systems. By adopting the categorical approach, the construction of reactive autonomic systems and the self-configuration work flows can be specified, proved and composed formally with preserving their properties. In future, we will work toward building a categorical framework to verify the correctness of the construction of reactive autonomic systems, based on the proposed categorical specification.

## Acknowledgements

We thank the Editor and the referee for their comments. Research of M. Zhu and J. Li is funded by the Shandong University of Technology grants 4041-416069 and 4041-417010. The support is greatly appreciated.

## Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

## References

[1] Carvalho, G., Cavalcanti, A. and Sampaio, A. (2016) Modelling Timed Reactive Systems from Natural-Language Requirements. *Formal Aspects of Computing*, **5**, 1-41.

[2] Kuang, H., Ormandjieva, O., Klasa, S. and Bentahar, J. (2010) A Formal Specification of Fault-Tolerance in Prospecting Asteroid Mission with Reactive Autonomie Systems Framework. *Proceedings of the* 21*st IEEE International Conference on Application-Specific Systems*, *Architectures and Processors*, Rennes, 7-9 July 2010, 99-106. https://doi.org/10.1109/ASAP.2010.5540769

[3] Zhu, M., Grogono, P. and Ormandjieva, O. (2017) Exploring Relationships between Syntax and Semantics of a Process-Oriented Language by Category Theory. *Proceedings of the* 8*th International Conference on Ambient Systems*, *Networks and Technologies*, Madeira, 16-19 May 2017, 241-248. https://doi.org/10.1016/j.procs.2017.05.342

[4] Zhu, M., Li, J., Fan, G.D. and Zhao, K.S. (2018) Modeling and Verification of Response Time of QoS-Aware Web Service Composition by Timed CSP. *Proceedings of the* 9*th International Conference on Emerging Ubiquitous Systems and Pervasive Networks*, Leuven, 5-8 November 2018, 48-55. https://doi.org/10.1016/j.procs.2018.10.148

[5] Seshia, S.A. (2007) Autonomic Reactive Systems via Online Learning. *Proceedings of the* 4*th International Conference on Autonomic Computing*, Florida, 11-15 June 2007, 30-39. https://doi.org/10.1109/ICAC.2007.10

[6] Litoiu, M., Solomon, B., Ionescu, D. and Mihaescu, M. (2007) A Real-Time Adaptive Control of Autonomic Computing Environments. *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research*, Richmond Hill, 22-25 October 2007, 124-136.

[7] Winskel, G. and Nielsen, M. (1995) Models for Concurrency. *Handbook of Logic in*

*Computer Science*, **4**, 1-148.

[8] Nielsen, M., Sassone, V. and Winskel, G. (1996) Models for Concurrency: Towards a Classification. *Theoretical Computer Science*, **170**, 297-348. https://doi.org/10.1016/S0304-3975(96)80710-9

[9] Hildebrandt, T.T. (2000) Categorical Models for Fairness: Completion vs Delay. *Proceedings of the First Irish Conference on the Mathematical Foundations of Computer Science and Information Technology*, Cork, 20-21 July 2000, 188.

[10] Mackworth, A.K. and Zhang, Y. (2003) A Formal Approach to Agent Design: Anoverview of Constraint-Based Agents. *Constraints*, **8**, 229-242. https://doi.org/10.1023/A:1025697810124

[11] Enguix, G.B. and Lopez, M.D.J. (2007) Agent-Environment Interaction in a Multi-Agent System: A Formal Model. *Proceedings of the GECCO Conference on Genetic and Evolutionary Computation*, London, 7-11 July 2007, 2607-2612.

[12] Goubault, E. and Mimram, S. (2010) Formal Relationships between Geometrical and Classical Models for Concurrency. *Proceedings of the Workshop on Geometric and Topological Methods in Computer Science*, Denmark, 11-15 January 2010, 77-109. https://doi.org/10.1016/j.entcs.2012.05.007

[13] Ormandjieva, O. and Quiroz, J. (2008) Methodology for Automatic Generation of Exhaustive Behavioral Models in Reactive Autonomic Systems. *Proceedings of the International Conference on Software Engineering Theory and Practice*, Florid, 7-10 July 2008, 95-104.

[14] Quiroz, J. (2007) Methodology for Automatic Generation of Behavioral Specification in Reactive Autonomic Systems. Master's Thesis, Concordia University, Montreal.

[15] Barr, M. and Wells, C. (2012) Category Theory for Computing Science. Prentice-Hall, Upper Saddle River.