Towards Specifying Reactive Autonomic Systems with a Categorical Approach: A Case Study

Heng Kuang, Olga Ormandjieva, Stan Klasa, Noorulain Khurshid, and Jamal Benthar

Computer Science & Software Engineering Department, Concordia University
1515 St. Catherine St. West, Montreal, Quebec, Canada H3G 1M8
h_kuang@cse.concordia.ca

Abstract. Software complexity is the main obstacle to further progress in the IT industry. One solution is the autonomic system with self-* properties. Formal methods are proven approaches to ensuring the correct operation of complex interacting systems. However, the current formal methods do not adequately address the problem of verifying two of the most important features of auto-nomic systems, namely emergent behavior and evolving behavior. Category Theory (CT) has recently been proposed as a formal framework to provide a structure for isolating the management of evolving specifications and the analysis of changes. We propose a formal framework based on CT in this paper to specify reactive autonomic systems. Our approach is illustrated with a NASA case study.

Keywords: Category Theory, autonomic system, software engineering, formal method, reactive system.

1 Introduction

Although software engineering methodology and programming language innovation have extended both the size and the complexity of computing systems, depending on those solutions alone will not get the industry through the present software complexity crisis, which is the main obstacle to its further progress. This is because the difficulty of managing massive and complex computing systems goes well beyond the capability of IT professionals. Software complexity is derived from the following: 1) The need to integrate several heterogeneous software environments into one co-operative computing system; 2) The rapid stream of changing, as well as conflicting, demands at runtime requiring a timely and decisive response; and 3) The difficulty in anticipating and designing all the interactions among the elements of unpredictable, diverse, and interconnected systems. One of the remaining solutions is autonomic systems with self-* properties that help to address software complexity through the use of technology to manage technologies, specifically by hiding low-level complexities from end-users [1].

Since 2001, several researchers [2, 3] have proposed definitions for an autonomic system following the original vision of Horn [4]. The core of an autonomic system is self-adaptation, including self-organization, which can be achieved by realizing self-configuration, self-healing, self-optimization, and self-protection.

R. Lee & N. Ishii (Eds.): Soft. Eng. Research, Manage. & Appli. 2009, SCI 253, pp. 119–134.
springerlink.com

Reactive systems are some of the most complex systems, because they: 1) involve concurrency; 2) have very strict timing requirements; 3) must be reliable; 4) involve both software and hardware components; and 5) are intelligent and increasingly heterogeneous. These systems can be more self-adaptive to their environment and more self-organized when they are equipped with autonomic features. However, the current formal approaches do not have an appropriate mechanism for specifying Reactive Autonomic Systems (RAS) which can simplify and enhance the experience of end-users by anticipating their needs in a complex, dynamic, and uncertain environment.

Category Theory (CT) is a relatively young branch of mathematics, which was originally designed to express various structural concepts for mathematical fields in a uniform way, and has been successfully extended to software engineering [5]. The management of the analysis of changes and of evolving specifications in an RAS requires a specification structure that can isolate those changes within a small number of components and analyze the impacts of a change on interconnected components. CT is proposed in this paper to provide that structure because of its rich body of theory to help analyze specifications and their interactions, but also because it is abstract enough to integrate various specification languages. Moreover, automation can be achieved using CT; for instance, the composition of several specifications can be automatically derived with some properties, such as co-completeness.

We therefore propose a categorical approach to specify RAS. The rest of this paper is organized as follows: section 2 describes the case study through which we illustrate our approach; section 3 briefly presents a perspective view of our framework; section 4 introduces the categorical specification and describes on the case study how the CT can be used for specifying autonomic behavior; section 5 provides an overview of related work; and section 6 presents our conclusions and outlines directions for future work.

2 Case Study

We have chosen the Prospecting Asteroid Mission (PAM) as our case study, which is an application of NASA's Autonomous Nano Technology Swarm (ANTS) mission architecture [6]. The PAM consists of 1,000 pico-spacecraft, which are organized into 10 specialist classes with highly maneuverable and configurable solar sails, with types of rulers, messengers, and workers (imaging, IR spectrometer, magnetometer, altimeter, etc.). The basic design elements are self-similar low-power, low-weight, and addressable components and systems that can operate fully autonomously, along with adaptable units for swarm demands and environmental needs. Through the concurrent operation of 10 to 20 sub-swarms, hundreds of asteroids may be explored during a mission traverse of asteroid belts. Fig. 1 shows a sample scenario of the PAM [7].

The PAM must fulfill the following asteroid survey requirements: 1) optimal science operations at every object such as search of appropriate trajectories that can enable efficient operation of workers' instruments, as well as concurrent operations among multiple objects such as asteroid detection and tracking; 2) ongoing evolution of strategies as a function of object characteristics; 3) no single point failure, and robustness with respect to minor or critical loss; and 4) a high level of autonomy as a group of the specialized workers. The PAM is designed for a systematic study of an entire population of elements and involves not only a smart spacecraft, but also a

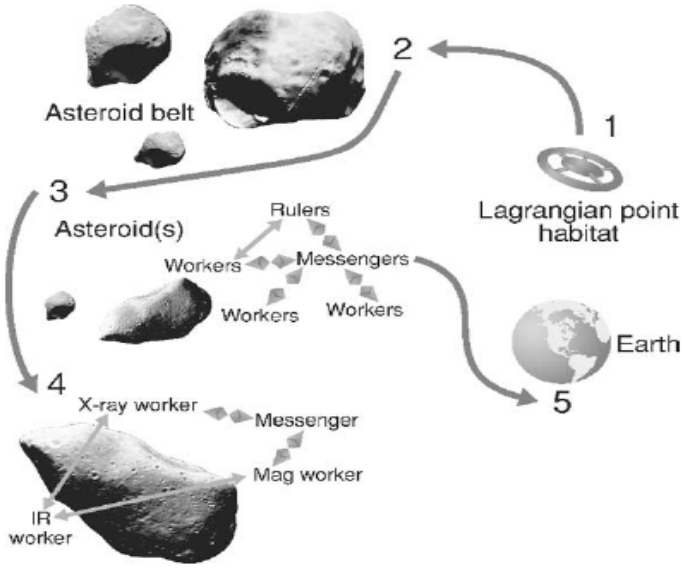


Fig. 1. A sample PAM scenario [7]

totally autonomic and distributed network of sensors or spacecraft with specialized device capabilities, for instance, computing, imaging, and spectrometry, as well as adaptable and evolvable heuristic systems. Furthermore, the sub-swarms of spacecraft can operate autonomously to enable optimal gathering of complimentary measurements for selected targets, and can also simultaneously operate in a broadly defined framework of goals to select targets from candidate asteroids [6].

The PAM spacecraft study a selected target by offering the highest quality and coverage of measurement by particular classes of measurers, called virtual teams. A virtual *instrument* team is made up of members of each class to optimize data collection. Another strategy involves providing comprehensive measurement to solve particular scientific problems by forming virtual *experiment* teams made up of multiple specialist classes, such as a dynamic modeler team, an asteroid detector and stereo mapper team, a petrologist team, a prospector team, a photogeologist team, etc. The social structure of the PAM swarm can be determined by a particular set of scientific and mission requirements, and representative system elements may include [8]: 1) a *general*, for distributed intelligence operations, resource management, mission conflict resolution, navigation, mission objectives, and collision avoidance; 2) *rulers*, for heuristic operation planning, local conflict resolution, local resource management, scientific discovery data sharing, and task assignment; 3) *workers*, for local heuristic operation planning, possible ruler replacement, and scientific data collection [8]. An operational scenario (see Fig. 1) is described as the following [8]:

1) Asteroid detection and tracking: the ANTS spacecraft travel through an asteroid belt; the workers with IR/Visible imaging devices continuously track asteroids, and information can be propagated if new asteroids are detected. The rulers de-cide which asteroids are of interest.

2) Ruler reaction: the rulers may assign degrees of importance to the asteroids, and then *hoverers* begin to observe the most important asteroids.
3) Arrival at asteroids: the *messengers* arrive before the workers to the vicinity of the asteroids, and have a better communication range, so they can act as communication nodes for other spacecraft and transfer data from the workers to Earth. In addition, simple models of the asteroids can be created and sent to the workers, which will help individual workers to plan their trajectories on those asteroids.
4) Worker acquisition of data: the workers arrive and search for appropriate trajec-tories to enable efficient operation of their instruments and to prevent collisions, which is important when those workers drop toward the asteroids. (In order to reduce the possibility of a single failure point, trajectory determination should be distributed among individual workers, rather than handled by a central controller. Moreover, those workers should be able to adapt their observation plans to take advantage of interesting features when they are detected.)
5) Worker completion of observations: the workers can either move away from the asteroids, or approach them. The spacecraft also needs time to reduce the raw data to make them suitable for transportation models and statistics; in addition, the workers may call the messengers and transfer the reduced data to those me-ssengers that move among the workers. If the messengers reach their memory limits, they move to Earth and download information to communication points. Finally, the workers move to the next important asteroid.

The PAM can therefore be regarded as an RAS with autonomic properties [7]. The resources can be configured and reconfigured to support parallel operations at hundreds of asteroids over a given period (self-configuration). For example, a sub-swarm may be organized for scientific operations at an asteroid, and this sub-swarm can be reorganized at another asteroid. The rulers may maintain data on different types of asteroids and determine their characteristics over time. Therefore, the whole system can be optimized because time will not be wasted on the asteroids that are not of interest or are difficult to observe (self-optimization). The messengers provide communication among the rulers, the workers, and Earth, and so they can adjust their positions to balance that communication (self-adaptation). The PAM individuals should be capable of coordinating their orbits and trajectories to avoid collisions with other individuals in a reactive way. Moreover, the plans of the rulers should in-corporate the constraints necessary for acceptable collision risk between the spacecraft when they perform observation tasks (self-protection and reactive). The rulers capable of sensing solar storms should invoke the goal of protecting their missions when they recognize a threat of such storms. In addition, the rulers can inform the workers of the potential for these events to occur, so that they can orient their solar panels and sails to minimize the impact of solar wind. The rulers can also power down the workers' subsystems to minimize the disruption from charged particles (self-protection and self-adaptation).

Our goal is to establish a formal framework, the Reactive Autonomic Systems Framework (RASF), to model the RAS. The first step in this paper is to build a formal specification of the RAS meta-model described in section 3.

3 Reactive Autonomic Systems Framework

RAS meta-modeling focuses on and supports the process of construction of RAS models by providing "correct by construction" rules, constraints and properties applicable and useful for modeling reactive autonomic systems; its main concern is to make them evolve. The RAS meta-model (see Fig. 2 below) is a four-layer meta-modeling architecture which consists of the Reactive Autonomic Objects (RAO), the Reactive Autonomic Components (RAC), the Reactive Autonomic Component Group (RACG), and the RAS. The autonomic features are implemented by the RAO Leaders (RAOL), the RAC Supervisors (RACS), and the RACG Managers (RACGM) at the RAC, RACG, and RAS layers respectively.

The instruments in a spacecraft, such as an IR device, a Mag device, or a Sail can be specified as an RAO. The RAC is modeled by a set of synchronously communicating RAO, where one of them is named team leader (RAOL). The team members are responsible for reactive tasks, and the RAOL works on autonomic tasks. Every spacecraft in the PAM, such as the messenger, ruler, or worker, may be specified as an RAC, and the control unit in that spacecraft as an RAOL. The RACG is a set of RAC that cooperate in the fulfillment of group tasks through synchronous communication, and it is the minimum reactive autonomic element that can independently complete a full reactive task in the RAS meta-model. The autonomic behavior at this layer is coordinated by a supervisor (RACS); each sub-swarm in the PAM can be modeled as an RACG, and the ruler in that sub-swarm is an RACS. The RAS is a set of RACG with their asynchronous communication, and can provide an integrated interface for users to delegate tasks, manage repositories, and monitor systems. A manager (RACGM) is mainly responsible for coordinating autonomic behavior at this

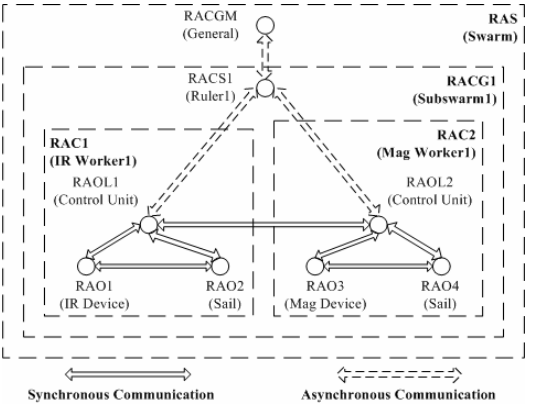


Fig. 2. An example of the PAM model conforming to the RAS meta-model

layer. The whole swarm in the PAM may be specified as an RAS, and the general in that swarm is an RACGM. Fig. 2 depicts an example of the RAS meta-model instantiation for the PAM case study.

The rationale for using CT to specify the RAS meta-model and the RAS categorical specification are presented in the following section.

4 Categorical Specification

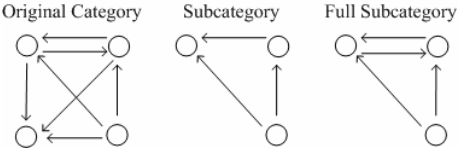
CT for software specification adopts the *correct by construction* approach, where components can be specified, proved, and composed so as to preserve their structures [9]. The term *diagram* [17] in CT takes its formal meaning and carries with it the intuition that comes from practice. Compared to other software concept formalizations, CT is not a semantic domain for formalizing the description of components or their connectors, but rather expresses the semantics of interconnection, configuration, instantiation, and composition, which are important aspects of modeling the evolving behavior of an RAS. Modeling can be achieved at a very abstract level, because CT proposes a toolbox which can be applied to any formalism for capturing component behavior, as long as that formalism satisfies certain properties of structure. Moreover, CT focuses on the relationships (morphisms) between objects, instead of on their representation. The morphisms may help determine the nature of the interactions established among the objects. Thus, a particular category may reflect a specific architectural style. CT can also provide the techniques for manipulating and reasoning on diagrams for building the hierarchies of system complexity, allow systems to be used as the components of more complex systems, and infer the properties of the systems from their configurations [10]. Let us recall some CT definitions [11] that will be used in this paper.

Definition 4.1. A category **C** consists of following data and rules:

- A class of *objects*: A, B, etc. We use $|\mathbf{C}|$ to denote the set of all objects, such as A, $B \in |\mathbf{C}|$.
- A class of *arrows (morphisms)*: f, g, etc.
- For each arrow f: $A \rightarrow B$, A is called the *domain* of f, denoted dom(f), and B is called the *codomain* of f, denoted cod(f). We use $\mathbf{C}(A,B)$ to indicate the set of all arrows in **C** from A to B.
- For each pair of arrows f: $A \rightarrow B$ and g: $B \rightarrow C$, a *composite morphism* is $g \circ f$: $A \rightarrow C$.
- For each object A, an *identity morphism* has both domain A and codomain A as Id_A: $A \rightarrow A$.
- Identity composition: $f \circ \mathrm{Id}_A = f = \mathrm{Id}_B \circ f$ for each morphism f: $A \rightarrow B$.
- Associativity: $h \circ (g \circ f) = (h \circ g) \circ f$ for each set of morphisms f: $A \rightarrow B$, g: $B \rightarrow C$, h: $C \rightarrow D$.
- Inverse of a morphism f: $A \rightarrow B$ is a morphism g: $B \rightarrow A$ such that $f \circ g = \mathrm{Id}_B$ and $g \circ f = \mathrm{Id}_A$; we denote the inverse of f as f^{-1} if it exists, and a morphism can have at most one inverse.

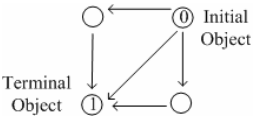
If f has an inverse, it is said to be an isomorphism; if f: $A \rightarrow B$ is an isomorphism, then A and B are said to be isomorphic, denoted as $A \cong B$. For example, the functions between sets give rise to a category, where the objects are sets and the morphisms are all functions between them. According to the RAS meta-model we presented in Section 3, the RAC can be specified by the category **RAC** having a set of objects (RAO) and their interactions as the morphism f: **RAC**(RAO_i, RAO_j) where RAO_i, $RAO_j \in$ |**RAC**|. Every spacecraft in the PAM, an IR worker, for instance, is a category **IR-Worker** consisting of objects *IR-Device*, *Sail*, *Control-Unit*, as well as their interactions **IR-Worker**(*IR-Device*, *Sail*), **IR-Worker**(*Control-Unit*, *IR-Device*), and **IR-Worker**(*Control-Unit*, *Sail*).

Definition 4.2. Let **C** and **D** be categories. **C** is a *subcategory* of **D** denoted as $\mathbf{C} \lhd \mathbf{D}$ if $|\mathbf{C}| \subseteq |\mathbf{D}|$, and the morphisms of **C** are morphisms of **D** as $\mathbf{C}(A_i, A_j) \subseteq \mathbf{D}(A_i, A_j)$ where $A_i, A_j \in |\mathbf{C}|$; **C** is a *full subcategory* of **D** when $\mathbf{C}(A_i, A_j) = \mathbf{D}(A_i, A_j)$ for all objects of **C**.

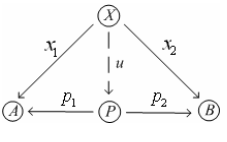


The RACG can be specified as a category **RACG** with a set of full subcategories **RAC**, and the RAS may be specified by the category **RAS** having a family of full subcategories **RACG**. Similarly, for a sub-swarm in the PAM, for example, a petrologist team is the category **Petrologist** including a set of full subcategories **Mag Worker**, **X-ray Worker**, and **Imaging Worker**.

Definition 4.3. In any category **C** an object: 1) 0 is *initial* if, for any object C, there is a unique morphism $\rightarrow C$, such as the empty set $\{\}$ in the category of sets; 2) 1 is *terminal* if, for any object C, there is a unique morphism $C \rightarrow$, such as for any singleton set in the category of sets; 3) initial (terminal) objects are unique up to isomorphism.

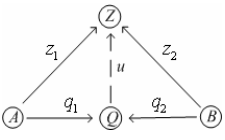


Definition 4.4. The *product of objects A* and *B, denoted by* $A \times B$, is an object P along with the arrows $A \xleftarrow{p_1} P \xrightarrow{p_2} B$, and P is the terminal object in the cate-gory of all such candidates X. A pair of objects may have many different products in a category, but those products are unique up to isomorphism, if they exist.

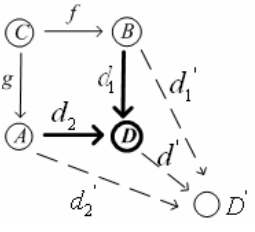


The synchronous communication between the RAO may be represented by their product. For example, the interaction between the *Mag Device* (*A*) and *Sail* (*B*) in a **Mag Worker**, can be specified by their synchronous product (*P*) where p_1, p_2 are state projections from *P* to the *Mag Device* and *Sail*; x_1, x_2 are state projections from all other candidates *X* to the *Mag Device* and *Sail.*

Similarly, a diagram $A \xrightarrow{q_1} Q \xleftarrow{q_2} B$ may be used to define *coproduct* (the dual concept), and *Q*, denoted by $A + B$, is the initial object in the category of all such candidates *Z*; coproducts are also unique up to isomorphism. An example of a co-product is the asynchronous communication among the Control Units of the worker, ruler, and general in the PAM where q_1, q_2 are messages from the Control-Units of an **IR-Worker** and **Mag-Worker** (*A*, *B*) to the Control-Unit of a ruler (*Q*); z_1, z_2 are messages from *A*, *B* to all other candidates of *Q*.

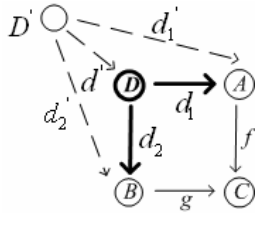


Definition 4.5. The *pushout* for two morphisms *f*: $C \rightarrow B$ and *g*: $C \rightarrow A$, denoted as $A +_C B$, is an object *D* along with two morphisms d_1: $B \rightarrow D$ as well as d_2: $A \rightarrow D$ such that the following diagram commutes, and *D* is the initial object in the category of all such candidates D'.



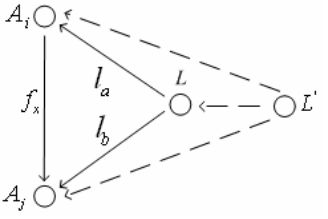
The pushout can be used to represent the next relay of outgoing communication from the same source object (*RAO*) as $RAO' = RAO_i +_{RAO} RAO_j$. For instance, a ruler (*C*) sends some instructions (*g*) to the *Control Unit* of an **X-ray Worker** (*A*) as well as instructions (*f*) to a **Alt Worker** (*B*), and the processing outcome (*d1'* and *d2')* from those two workers will be integrated and transmitted (*d'*) to the general or ground station (D') by a messenger (*D*).

Dually, the *pullback* for two morphisms, denoted as $A \times_C B$, can be defined as in the following diagram, and *D* is the terminal object in the category of all such candidates D'.

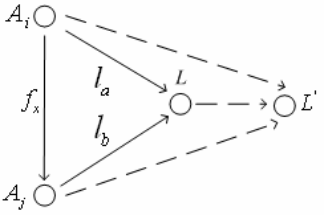


The pullback may represent the previous relay of incoming communication toward the same destination *RAO* as $RAO' = RAO_i \times_{RAO} RAO_j$. For instance, a messenger (*D*) forwards some instructions (d_1, d_2) from a ruler (D') to the Control Units of an **Imaging Worker** (*A*) and a **Gamma-ray Worker** (*B*), and the working outcome from those two workers (*f*, *g*) will be sent to another ruler or messenger (*C*).

Definition 4.6. For any diagram containing objects A_i along with morphisms f_i, the *limit* of this diagram is an object *L* together with a set of morphisms *l*, such that, for each $l_a: L \to A_i$, $l_b: L \to A_j$, $f_x: A_i \to A_j$, then $f_x \circ l_a = l_b$, and *L* is the terminal object in the category of all such candidates L', as the following diagram illustrates.

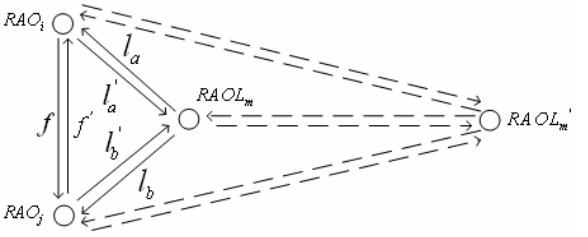


Dually we have the concept of the *colimit*, which is an object *L* along with a set of morphisms *l*, such that for each $l_a: A_i \to L$, $l_b: A_j \to L$, $f_x: A_i \to A_j$, then $l_b \circ f_x = l_a$, and *L* is the initial object in the category of all such candidates L' as depicted in the following diagram.



If we start with a diagram of the *RAO*, a kind of universal communicator may be introduced, and this is a higher-level object with arrow connections to each object in a base diagram. Thus, we can model that object as a limit or colimit of the base diagram. Graphically speaking, the limit object is a domain of all the arrows going to the *RAO* in the base diagram, and the colimit object is a codomain of all the arrows coming from the *RAO* in the base diagram. Having the limit or colimit object allows for the modeling of each specific interaction between the *RAO* by the communication path from the limit or colimit object to those *RAO*. According to the definition of the limit and colimit, no other object in the diagram above can improve the communication capability of the limit and colimit object due to the commutativity constraint in the universal properties of a limit and colimit.

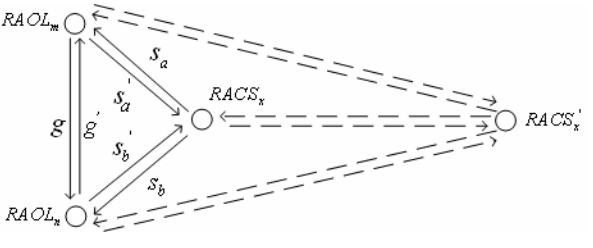
Because the RAC is represented as a category of the RAO, its behavior is derived from those RAO and can be specified by their limit or colimit. Thus, the interactions among the RAO (*f*, *f'*) may be interpreted as the incoming (l_a, l_b) or outgoing (l_a', l_b') communication between those RAO and their leader (RAOL), as shown in the following diagram. In the PAM, the behavior of a spacecraft, such as an **IR Worker**, can be represented by the behavior of its *Control Unit* that is specified as the limit or colimit of the *IR Device* and *Sail*.



As a result, the grid-like communication among the RAO can be regarded as the cone-like communication between those RAO and their RAOL, by converting their relationship of many-to-many to one-to-many or many-to-one through a categorical computation. Such model facilitates the specification of the emergent behavior of those RAO by hiding the many-to-many relationship details.

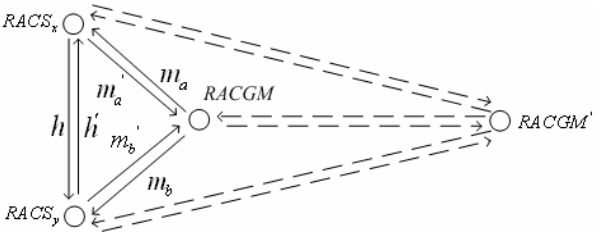
Because the behavior of an RAC may be described as the limit or colimit (RAOL) of its RAO, the RACG can be specified by the category **RACG** having a set of objects (RAOL) and their interactions (g, g') as the morphism f: **RACG**($RAOL_m$, $RAOL_n$), where $RAOL_m$, $RAOL_n \in$ |**RACG**|. A sub-swarm in the PAM, for example a prospector team, is the category **Prospector** including the *Control Units* of its **X-ray Worker** ($RAOL_m$), **Mag Worker** ($RAOL_n$), and their interactions (g, g'), such as **Prospector** (*Control Unit*$_{Alt}$, *Control Unit*$_{Mag}$).

Because the behavior of an RACG is derived from its RAC, the limit or colimit of those RAOL ($RACS_x$) may be used to specify the behavior of the RACG. Thus, the communication among the RAOL (g, g') can be interpreted by the incoming or outgoing interactions between those RAOL (s_a, s_b, s_a', s_b') and their supervisor (RACS), as depicted in the following diagram. For instance, the behavior of a sub-swarm in the PAM, such as a photogeologist team, may be represented by the behavior of its ruler that is specified as a limit or colimit of the *Control Units* from its **Imaging Worker** and **Alt Worker**.



Similarly, the RAS can be specified by the category **RAS** with a set of objects (RACS) and their interaction (h, h') as the morphism f: **RAS**($RACS_x$, $RACS_y$), where $RACS_x$, $RACS_y \in$ |**RAS**|. Thus, the whole swarm in the PAM is the category **PAM-SWARM**, having the rulers of its asteroid detector and stereo mapper team, petrologist team, photogeologist team, prospector team, dynamic modeler team, and their interactions; for instance, **PAM-SWARM**($ruler_{modeler}$, $ruler_{photo}$). As the behavior of the RAS is derived from its RACG, the limit or colimit of those RACS may be used to represent the behavior of the RAS. Thus, the communication among the RACS (h, h') can be modeled as the incoming or outgoing interactions between those RACS and their manager (RACGM) (m_a, m_b, m_a', m_b'), as illustrated in the following diagram. For

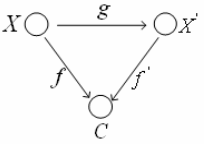
example, the behavior of the whole swarm in the PAM may be represented by the behavior of its general, which is specified as a limit or colimit of the rulers from its sub-swarms, such as $ruler_{mapper}$, $ruler_{petro}$, $ruler_{prospector}$, etc.



If we consider a category where objects are morphisms, a slice (coslice) category can be defined.

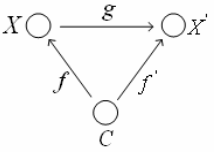
Definition 4.7. A *slice category* **C**/*C* of a category **C** over its object $C \in |\mathbf{C}|$ (sometimes called a *comma category*) has the following data:

- A class of objects $f \in \mathbf{C}$ such that cod(f) = C.
- A class of arrows g from f: $X \to C$ to $\dot{f}$: $\dot{X} \to C$ such that $\dot{f} \circ g = f$.



The outgoing communication from the RAO to its RAOL in an RAC may be specified by a slice category as **RAC**/$RAOL_m$, where each object is the outgoing communication $(f, \dot{f})$ and the morphism is the arrow g from f: $RAO_i(X) \to RAOL_m(C)$ to $\dot{f}$: $RAO_j(\dot{X}) \to RAOL_m(C)$ such that $\dot{f} \circ g = f$. Similarly, the outgoing communication from the RAOL and RACS to the RACS and RACGM can be represented by their slice categories as **RACG**/$RACS_x$ and **RAS**/*RACGM*. The outgoing communication in a spacecraft from its instruments to their control unit is a slice category, for instance, **Imaging Worker**/*Control Unit*; the outgoing communication between the spacecraft and the messenger in a sub-swarm is a slice category, such as **Petrologist**/*messenger*.

Dually, a *coslice category* *C*/**C** has objects $f \in \mathbf{C}$ such that dom(f) = C and arrows from f: $C \to X$ to $\dot{f}$: $C \to \dot{X}$ such that $g \circ f = \dot{f}$.



The incoming communication from the RAOL to the RAO within an RAC can be specified by a coslice category as $RAOL_m$/**RAC**, where objects are incoming communication $(f, \dot{f})$ and the morphism is an arrow g from f: $RAOL_m(C) \to RAO_i(X)$ to $\dot{f}$: $RAOL_m(C) \to RAO_j(\dot{X})$ such that $g \circ f = \dot{f}$. Similarly, the coslice categories can be used to represent the incoming communication from the RACS and RACGM to the

RAOL and RACS as $RACS_x$/ **RACG** and *RACGM*/**RAS** respectively. As a result, the incoming communication in a sub-swarm from its ruler to the control unit of each spacecraft is a coslice category, such as *ruler*/**Prospector**. Also, the incoming communication in the whole swarm from the general to the rulers is a coslice category *general*/**PAM-SWARM**.

Considering a category where objects are categories and morphisms are mappings between those categories, the morphisms in that category are called functors.

Definition 4.8. A *functor* ("the homomorphism of categories") F: $\mathbf{C} \rightarrow \mathbf{D}$ between two categories **C** and **D** is a mapping of objects to objects along with arrows to arrows from **C** to **D** in the following way:

- Object mapping as F: $|\mathbf{C}| \rightarrow |\mathbf{D}|$.
- Arrow mapping as F: $\mathbf{C}(A_i, A_j) \rightarrow \mathbf{D}(F(A_i), F(A_j))$.
- Composition mapping as $F(g \circ f) = F(g) \circ F(f)$ where $g, f \in \mathbf{C}$ and $F(g), F(f) \in \mathbf{D}$.
- Identity mapping: $F(\mathrm{Id}_A) = \mathrm{Id}_{F(A)}$ where $\mathrm{Id}_A \in \mathbf{C}$ and $\mathrm{Id}_{F(A)} \in \mathbf{D}$.

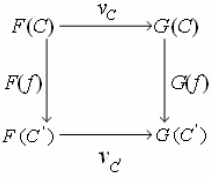
The evolution of an RAC, because of self-adaptation and self-organization during run time, can be represented by functors. For instance, the evolution from the RAC to RAC' is a functor F, which includes a mapping of objects (*RAO*) in **RAC** to the objects (*RAO*') in **RAC**' (F: $|\mathbf{RAC}| \rightarrow |\mathbf{RAC}'|$), as well as a mapping of the morphisms (interactions among the *RAO*) in **RAC** to morphisms (interactions among the *RAO*') in **RAC**' (F: $\mathbf{RAC}(RAO_i, RAO_j) \rightarrow \mathbf{RAC}'(F(RAO_i), F(RAO_j))$). Similarly, the evolution of the RACG and RAS may be represented as F: $\mathbf{RACG} \rightarrow \mathbf{RACG}'$ and F: $\mathbf{RAS} \rightarrow \mathbf{RAS}'$ respectively. The evolution of a spacecraft in the PAM, for example, from **Alt Worker** to **Alt Worker**', because of the new configuration for its *altimeter* or *sail*, can be specified by a functor as F: **Alt Worker** $\rightarrow$ **Alt Worker**'; moreover, the evolution of a sub-swarm, for instance, from the **Photogeologist** to **Photogeologist**' due to the new organization for its **Imaging Worker** or **Alt Worker** may be modeled as F: **Photogeologist** $\rightarrow$ **Photogeologist**'.

Definition 4.9. The *product of categories* **C** and **D**: $\mathbf{C} \times \mathbf{D}$ has objects of the form (C, D), where $C \in |\mathbf{C}|$, $D \in |\mathbf{D}|$, along with arrows of the form (f, g): $(C, D) \rightarrow (C', D')$, where f: $C \rightarrow C' \in \mathbf{C}$ and g: $D \rightarrow D' \in \mathbf{D}$. Both the unit and composition are defined as: $1_{(C, D)} = (1_C, 1_D)$, $(f', g') \circ (f, g) = (f' \circ f, g' \circ g)$, and there are two *projection functors*: $\mathbf{C} \xleftarrow{\pi_1} \mathbf{C} \times \mathbf{D} \xrightarrow{\pi_2} \mathbf{D}$ defined by $\pi_1(C, D) = C$, $\pi_1(f, g) = f$, and similarly for π_2.

For example, the interaction between the RAC can be specified as a product of two categories $\mathbf{RAC}_m \times \mathbf{RAC}_n$, which has objects of the form (RAO_m, RAO_n) for $RAO_m \in |\mathbf{RAC}_m|$, $RAO_n \in |\mathbf{RAC}_n|$, along with arrows of the form (f, g): $(RAO_m, RAO_n) \rightarrow (RAO_m', RAO_n')$ for f: $RAO_m \rightarrow RAO_m' \in \mathbf{RAC}_m$ and g: $RAO_n \rightarrow RAO_n' \in \mathbf{RAC}_n$. Similarly, the interaction between the RACG may be specified by a product of two categories as $\mathbf{RACG}_x \times \mathbf{RACG}_y$. The interaction between two spacecraft, such as **IR Worker** and **Alt Worker**, can be represented by **IR Worker**$\times$**Alt Worker** having objects of the form ($IR\ Control\text{-}Unit_{state0}$, $Alt\text{-}Control\text{-}Unit_{state0}$) and morphisms of the form ($IR\text{-}Control\text{-}Unit_{state0}$, $Alt\text{-}Control\text{-}Unit_{state0}$)$\rightarrow$($IR\text{-}Control\text{-}Unit_{state1}$, $Alt\text{-}Control\text{-}Unit_{state1}$).

In a category where objects are functors, mappings between the functors are called natural transformations.

Definition 4.10. For categories **C**, **D**, along with functors F, G: **C** → **D**, a *natural transformation* (v: $F \rightarrow G$) is a family of arrows in **D** as v_c: $F(C) \rightarrow G(C)$, such that, for any f: $C \rightarrow C'$ in **C**, $v_{C'} \circ F(f) = G(f) \circ v_C$, as in the following diagram. Given such a natural transformation v, the **D**-arrow v_C is called a *component* of v at C, and, if v is invertible, it is known as a *natural isomorphism*.



For example, every group is naturally isomorphic to its opposite group. Because the evolutions of the RAC, RACG, and RAS are specified as functors from category **RAC** to **RAC**', **RACG** to **RACG**', and **RAS** to **RAS**', the natural transformation may represent the mapping of those alternative evolutions. The relationship between two solutions in terms of fixing a problem for a sub-swarm, *Solution1*: **Prospector** → **Prospector**' and *Solution2*: **Prospector** → **Prospector**", can be modeled by a natural transformation *convert*: *Solution1* → *Solution2*.

Definition 4.11. A *functor category* Fun(**C**, **D**) has:

- Objects: functors F: **C** → **D**.
- Arrows: natural transformations v: $F \rightarrow G$.
- For each object F, 1_F has components $(1_F)_C = 1_{FC}$: $FC \rightarrow FC$, and composite of $F \xrightarrow{\alpha} G \xrightarrow{\beta} H$ has components $(\beta \circ \alpha)_C = \beta_C \circ \alpha_C$.

All possible evolutions, along with their relationships for the RAC, RACG, and RAS, can be specified as functor categories Fun(**RAC**, **RAC**'), Fun(**RACG**, **RACG**'), and Fun(**RAS**, **RAS**') respectively. For example, all the plans to solve a problem for a spacecraft and their relations, such as **Petrologist**, may be represented by the functor category Petro-Fun(**Petrologist**, **Petrologist**').

In an abstract sense, we are dealing with arrow diagrams, where the objects are RAO, RAOL, RACS, and RACGM, and arrows are communication channels among those objects or groups of objects (RAC, RACG, and RAS). Moreover, commutativity can be interpreted in a natural way, that communication paths yield the same result, and we may also obtain some categorical properties of the RAS meta-model from its categorical specification above.

Property 4.1. Isomorphic objects interact in the same way. Accordingly, the *RAO* can be replaced by an isomorphic one (*RAO*') through the isomorphism and its inverse to re-establish the interaction as the following: any incoming arrow from RAO_i to *RAO* ($RAO \xleftarrow{g} RAO_i$) is replaced by $RAO' \xleftarrow{g \circ f} RAO_i$; and any outgoing arrow from *RAO* to RAO_i ($RAO \xrightarrow{h} RAO_i$) is replaced by $RAO' \xrightarrow{f^{-1} \circ h} RAO_i$. This property

may be employed in a fault-tolerance mechanism to formally model self-healing behavior and verify the consistency of the replaced RAO with that of the faulty RAO. For instance, in order to take over a crashed spacecraft in the PAM, the substitute should be isomorphic to the original.

Property 4.2. A category is called finitely complete (cocomplete) if all the finite diagrams in that category have limits (colimits). Therefore, at least one *RAOL*, *RACS*, and *RACGM* (limit or colimit) is required in the category **RAC**, **RACG**, and **RAS** respectively to ensure the completeness (cocompleteness) of the RAC, RACG, and RAS. This property means that, no matter how those RAC, RACG, and RAS evolve due to their self-adaptation and self-organization during run time, the fulfillment of designated tasks, behavior, and communication must be preserved and verified. This property may be employed in self-configuration to ensure the completeness and co-completeness of a new configuration while the RAS is evolving; for example, at least one control unit for every spacecraft, one ruler for each sub-swarm, and one general for the whole swarm in the PAM.

Fig. 3 illustrates the categorical specification on the PAM configuration depicted in Fig. 2.

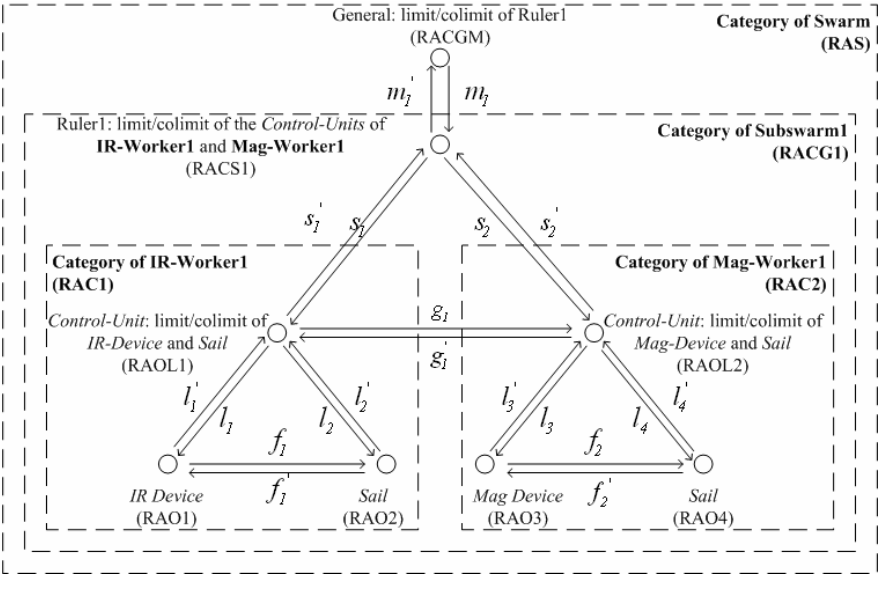


Fig. 3. Categorical specification of PAM Model

5 Related Work

The only published work on modeling autonomous systems using CT [12] served as the structure for the research presented in this paper. Its author stated that an auto-nomous system is a set of cooperating subsystems, and defined a specification

lan-guage for such systems based on CT. A constructor for communication by using monoids was introduced, and the feasibility of the categorical approach was proven, but no systematic methodology was proposed. There is also some related work regarding our case study. The paper [13] states a formal task-scheduling approach and model the self-scheduling behavior of the ANTS by an autonomic system specification language. The authors in [14] summarize necessary properties for the effective specification and emergent behavior predication of the PAM. They also compared current formal methods and integrated formal methods for the specification of intelligent swarm systems with the emergent behavior.

However, there is no single formal method satisfying all the required properties for specifying the PAM, and the PAM specification cannot be easily converted to program code or used as the input for model checkers when using integrated formal methods. Our research considerably differs from the related work above, since our goal is to propose a systematic and formal methodology based on CT to model the RAS, which could be implemented by multi-agent systems (MAS), service-oriented systems, or object-oriented systems. Our categorical approach is abstract enough to accommodate various specification languages, and it also proposes a toolbox for the formalisms to capture component behavior in the PAM, as long as those formalisms satisfy certain structural properties. If we consider a category in which objects are specifications, the morphisms in that category will translate the vocabulary of one specification into another while preserving the theorems.

6 Conclusions and Future Work

This paper introduced an important direction with respect to the formal aspects of modeling the RAS using CT. The work is motivated by the importance of a compliance with the self-management requirements for increasingly complex RAS. Our formal approach employs CT as a unified formal language allowing the use of the same constructors to model heterogeneous objects and the various types of relations between them. We have shown that CT is expressive enough to capture the knowledge about the RAS constructs, along with their interrelations, in a single formal representation in which structure and reasoning are bound together.

We are currently working on a formal specification of the Categorical Modeling Language (CML), which can be used to present the categorical specification and self-* properties for specifying autonomic behavior, and a graphical tool to capture RAS modeling. Once the RAS meta-model has been developed, we will transfer it to the MAS model, since an agent-based approach is considered a natural way to model the RAS [15]. The RAOL, RACS, and RACGM can be modeled as the hybrid agents [16]. Finally, a source code template can be generated according to the MAS model, and this will be discussed in our future work.

References

1. IBM Corporation, An Architectural Blueprint for Autonomic Computing, White Paper, 4th edn (June 2006)
2. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. Computer 36(1), 41–50 (2003)

3. Lin, P., MacArthur, A., Leaney, J.: Defining Autonomic Computing: A Software Engineering Perspective. In: Proceedings of the 16th Australian Software Engineering Conference, Brisbane, Australia, March 2005, pp. 88–97 (2005)
4. Horn, P.: Autonomic Computing: IBM Perspective on the State of Information Technology. Presented at AGENDA 2001, October 2001, IBM T. J. Watson Labs (2001)
5. Fiadeiro, J.: Categories for Software Engineering. Springer, Heidelberg (2004)
6. Clark, P.E., Rilee, M.L., Truszkowski, W., Marr, G., Curtis, S.A., Cheung, C.Y., Rudisill, M.: PAM: Biologically Inspired Engineering and Exploration Mission Concept, Components, and Requirements for Asteroid Population Survey. In: Proceedings of the 55th International Astronautical Congress, Vancouver, Canada (October 2004) IAC-04-Q5.07
7. Truszkowski, W.F., Hinchey, M.G., Rash, J.L., Rouff, C.A.: Autonomous and Autonomic Systems: a Paradigm for Future Space Exploration Missions. IEEE Transaction on Systems, Man, and Cybernetics, Part C: Applications and Reviews 36(3), 279–291 (2006)
8. Curtis, S., Mica, J., Nuth, J., Marr, G., Rilee, M., Bhat, M.: ANTS (Autonomous Nano Technology Swarm): an Artificial Intelligence Approach to Asteroid Belt Resource Exploration. Proceedings of the 51st International Astronautical Congress (October 2000) IAA-00-IAA.Q.5.08
9. Wiels, V., Easterbrook, S.: Management of Evolving Specifications Using Category Theory. In: Proceedings of the 13th IEEE International Conference on Automated Software Engineering, October 1998, pp. 12–21 (1998)
10. Fiadeiro, J.L., Maibaum, T.: A Mathematical Toolbox for the Software Architect. In: Proceedings of the 8th International Workshop on Software Specification and Design, Schloss Velen, Germany, March 1996, pp. 46–55 (1996)
11. Awodey, S.: Category Theory. Oxford University Press, USA (2006)
12. Lee, W.M.: Modelling and Specification of Autonomous Systems using Category Theory. PhD Thesis, University College of London, London, UK (October 1989)
13. Vassev, E., Hinchey, M., Paquet, J.: A Self-Scheduling Model for NASA Swarm-Based Exploration Missions Using ASSL. In: Proceedings of the 5th IEEE Workshop on Engineering of Autonomic and Autonomous Systems, Belfast, Northern Ireland, March 2008, pp. 54–64 (2008)
14. Hinchey, M.G., Rouff, C.A., Rash, J.L., Truszkowski, W.F.: Requirements of an Integrated Formal Method for Intelligent Swarms. In: Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems, Lisbon, Portugal, September 2005, pp. 125–133 (2005)
15. Tesauro, G., Chess, D.M., Walsh, W.E., Das, R., Segal, A., Whalley, I., Kephart, J.O., White, S.R.: A Multi-Agent Systems Approach to Autonomic Computing. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems, New York, NY, USA, July 2004, pp. 464–471 (2004)
16. Kuang, H., Ormandjieva, O.: Self-Monitoring of Non-Functional Requirements in Reactive Autonomic System Framework: A Multi-Agent Systems Approach. In: Proceedings of the 3rd International Multi-Conference on Computing in the Global Information Technology, Athens, Greece, July 2008, pp. 186–192 (2008)
17. Pfalzgraf, J.: On an Idea for Constructing Multiagent Systems (MAS) Scenarios. In: Advances in Multiagent Systems, Robotics and Cybernetics: Theory and Practice, IIAS, Tecumseh, ON, Canada, vol. 1 (2006)