ASSL Specification and Code Generation of Self-Healing Behavior for NASA Swarm-Based Systems*

Emil Vassev[1] and Mike Hinchey[2]
[1]Lero–the Irish Software Engineering Research Center, University College Dublin, Ireland
[2]Lero–the Irish Software Engineering Research Center, University of Limerick, Ireland
emil.vassev@lero.ie, mike.hinchey@lero.ie

Abstract

The Autonomic System Specification Language (ASSL) is a framework for formally specifying, validating and generating autonomic systems. This paper presents concrete results on the use of ASSL to specify a self-healing behavior model for NASA swarm-based exploration missions and to generate an application skeleton of the same. We present the specification and experiment with the generated code to demonstrate that ASSL generates operational code that is capable of self-management in respect of the specified self-healing model.

1. Introduction

Since its first announcement in 2001, Autonomic Computing [1] has inspired a tremendous number of initiatives for self-management of complex systems. Such an initiative is ASSL [2, 7], where we approach the problem of formal specification, validation, and code generation of autonomic systems [8, 9, 10] within a single framework.

Being an autonomic system (AS), the NASA Autonomous Nano Technology Swarm (ANTS) [3] concept mission follows the principles of autonomic computing [1] and provides self-management properties to ensure appropriate behavior and quality in the face of changing configurations and external conditions, based on automatic problem-determination algorithms.

In the course of this research, we applied ASSL to specifying a self-healing behavior model for ANTS and consecutively to generate an operational Java application skeleton of the same. Note that although operational, the code generated by the ASSL framework is a skeleton; i.e., some parts are generated as empty methods and classes. The results presented here are produced with the generated code only; i.e., without any additional implementation.

2. Related Work

A NASA developed formal approach, named R2D2C (Requirements to Design to Code) is described in [4]. In this approach, system designers may write specifications as scenarios in constrained (domain-specific) natural language, or in a range of other notations (including UML use cases). These scenarios are then used to derive a formal model that fulfills the requirements stated at the outset, and which is subsequently used as a basis for code generation. R2D2C relies on a variety of formal methods to express the formal model under consideration. The latter can be used for various types of analysis and investigation, and as the basis for fully formal implementations as well as for use in automated test case generation.

IBM has developed a framework called Policy Management for AC (PMAC) [5] that provides a standard model for the definition of policies and an environment for the development of software objects that hold and evaluate policies. For writing and storing policies, PMAC uses a declarative XML-based language called AC Policy Language (ACPL) [5, 6]. A policy written in ACPL provides an XML specification defining the following elements:

- condition - when a policy is to be applied;

* This work is funded in part by Science Foundation Ireland grant 03/CE02/I303_1 to Lero—the Irish Software Engineering Research Centre.

978-0-7695-3623-1/09 $25.00 © 2009 IEEE
DOI 10.1109/EASe.2009.12

- decision - observable behavior or desired outcome;
- result - a set of named and typed data values;
- action - invokes an operation;
- configuration profile - unifies result and action;
- business value - the relative priority of a policy;
- scope - the subject of the policy.

The basis of ACPL is the AC Expression Language (ACEL) [5, 6]. ACEL is an XML-based language developed to describe conditions when a policy should be applied to a managed system.

3. NASA Swarm-Based Missions

The Autonomous Nano Technology Swarm (ANTS) concept sub-mission PAM (Prospecting Asteroids Mission) is a novel approach to asteroid-belt resource exploration. ANTS provides extremely high autonomy, minimal communication requirements to Earth, and a set of very small explorers with a few consumables [3]. The explorers forming the swarm are pico-class, low-power, and low-weight spacecraft units, yet capable of operating as fully autonomous and adaptable agents.

There are three classes of ANTS spacecraft: *rulers*, *messengers* and *workers*. By grouping them in certain ways, ANTS forms teams that explore particular asteroids. The internal organization of a team depends on the task to be performed and on the current environmental conditions. In general, each team has a group leader (ruler), one or more messengers, and a number of workers carrying a specialized instrument. The messengers are needed to connect the team members when they cannot connect directly.

4. ASSL Multi-Tier Specification Model

The Autonomic System Specification Language [2, 7] is defined through formalization tiers. Over these tiers, ASSL provides a multi-tier specification model that is designed to be scalable and exposes a judicious selection and configuration of infrastructure elements and mechanisms needed by an AS. ASSL defines the latter with interaction protocols and autonomic elements (AEs), where the ASSL tiers and their sub-tiers describe different aspects of the AS under consideration, such as *policies*, *communication interfaces*, *execution semantics*, *actions*, etc.

By their virtue, the ASSL tiers and their sub-tiers (cf. Figure 1) are abstractions of different aspects of the autonomic system under consideration.

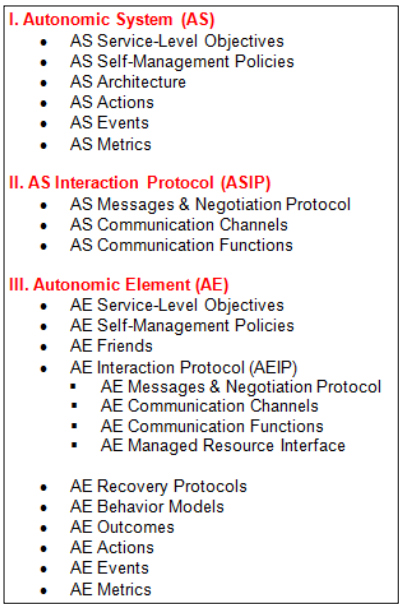

Figure 1. ASSL Tiers [7]

The AS Tier specifies an AS in terms of service-level objectives (AS SLO), self-management policies, architecture topology, actions, events, and metrics. The AS SLO is a high-level form of behavioral specification that establishes system objectives such as performance. The self-management policies could be the four self-management policies (the so-called self-CHOP) of an AS: *self-configuring*, *self-healing*, *self-optimizing*, and *self-protecting*, or they could be others. The metrics constitute a set of parameters and observables controllable by the AEs.

At the AS Interaction Protocol tier, the ASSL framework specifies an AS-level interaction protocol (ASIP). ASIP is a public communication interface, expressed with channels, communication functions and messages.

At the AE Tier, the ASSL formal model considers AEs to be analogous to software agents able to manage their own behavior and their relationships with other AEs. In this tier, ASSL describes the individual AEs of the AS.

5. Self-Healing Specification for ANTS

In ANTS, self-healing is about recovering from failures, including those caused by damage due to a

crash or any outside force. In our scenario, we assume that each *worker* sends, on a regular basis, *heartbeat messages* to the *ruler*. The latter can use those messages to determine when a worker is not able to continue its operation, due to a crash or a malfunction in its communication device. Moreover, a worker sends a *notification message* to the ruler if its instrument is malfunctioning or it has been broken, due to a collision with an asteroid or another spacecraft.

Thus, a ruler is notified in two ways for a worker loss:

- a heartbeat message from the worker has not been received;
- a message from the worker, notifying that an instrument is broken, has been received.

Once the loss of an operational unit has been detected, the ruler checks if the number of workers is below the critical minimum, and if so, it requests a replacement from another ruler. If such a replacement is not possible it could notify the ground control on Earth of the situation and request a replacement or further instructions from there (although in reality, this scenario is unlikely).

An ASSL specification of the ANTS self-healing behavior requires a specification at the AS tier for the global ANTS behavior, and at the AE tier for the self-healing behavior of every *worker* and *ruler*. Appendix A presents the ASSL self-healing specification model for ANTS. Note that, due to space limitations some of the specification structures are partially specified (cf. Appendix A for the specification of communication functions, channels and metrics). Due to the same reason, we specified only two AEs — a *ruler* (ANT_Ruler) and a *worker* (ANT_Worker).

In order to specify the self-healing autonomic property of a worker, we use the SELF_HEALING self-management policy specified at both the AS tier and AE tiers (*worker* and *ruler*). Note that the self-healing specification at the AS-tier (swarm level) handles only situations when a spacecraft unit has been lost (cf. Appendix A).

The SELF_HEALING policy is specified with a set of *fluents* and *mappings*, where the latter map the *fluents* to ASSL *actions*. In addition, we specify the necessary ASSL *actions*, *events* and *metrics*. Moreover, three interaction protocols (ASIP, and for each AE an AEIP) are specified to handle the communication between the AEs. Those protocols specify the messages to be exchanged among the *worker* and its *ruler*, the *communication functions*, and two *communication channels* - LBW_link and HBW_link (cf. Appendix A).

At the AEIP tiers, for both AEs we specify a managed element, which provides a getDistanceToNearestObject interface function. The latter is needed by the distanceToNearestObject metric to measure the distance to the nearest object.

Finally, note that the ANT_Ruler is listed as a *friend* [2] by the ANT_Worker, and thus, it can use the ANT_Worker's AEIP messages and channels.

6. Code Generation

In this section, we discuss the code generation results in terms of run-time self-management behavior. The results presented here were obtained by evaluating the successfully generated code for the ASSL self-healing model for ANTS.

6.1. Code Generation Statistics

ASSL groups the generated Java classes for an ASSL specification into hierarchically ordered Java packages. Figure 2 shows the packages of the Java application skeleton generated for the ANTS self-healing specification.

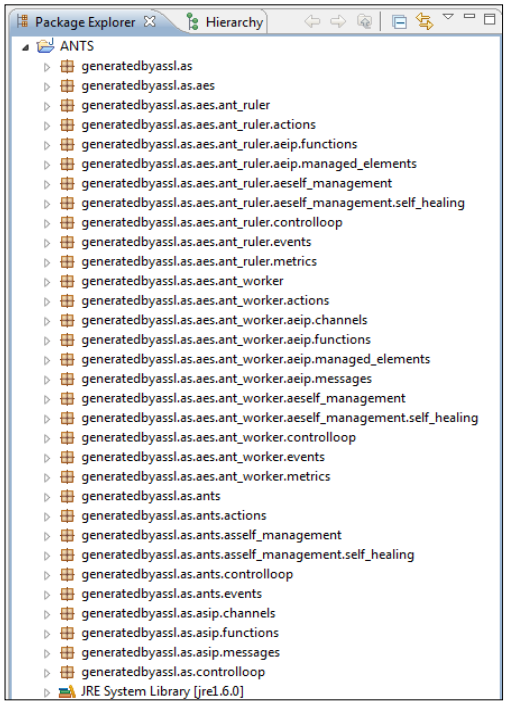

Figure 2. Generated Java Packages for ANTS

The ASSL framework generated 93 Java files for this specification (one per generated class or interface), which were distributed by the framework into 32 Java packages (cf. Figure 2). The total number of generated lines of code including comments was 8159. Compared to the ASSL self-healing specification model for ANTS, with 293 lines of ASSL code, we specified the self-healing policy at three levels—the AS tier level, the ANT_Worker AE level, and the ANT_Ruler AE level (cf. Appendix A). Therefore, the efficiency ratio in terms of lines of code (Java generated code versus ASSL specification code) is:

28 ≈ 8159 / 293

Thus, the ASSL code is significantly shorter, and hence more comprehensible, as one would expect in the case of an appropriate specification language for the domain.

6.2. Self-Healing Behavior

In this experiment, we experimented with the Java generated code for the ASSL self-healing specification model for ANTS (cf. Section V). Note that by default, all Java application skeletons generated with the framework generate run-time log records. The latter show important state-transition operations ongoing in the system at runtime. Thus, we can easily trace the behavior of the generated system by following the log records generated by the same.

Here we evaluated the log records produced by the generated Java application skeletons for three different versions of the ASSL self-healing specification model for ANTS. Thus, we modified the original version of the ANTS self-healing model to explore all aspects of the specified and generated self-healing behavior.

The following subsections present test experiments performed with the code generated for three different versions of the ASSL self-healing specification model for ANTS.

Test #1: Original Specification. In this test, we generated the Java application skeleton for the original ASSL self-healing specification model for ANTS (cf. Appendix A), compiled the same with Java 1.6.0, and ran the compiled code. The application ran smoothly with no errors.

First, it started all system threads as it is shown in the following log records. Note that starting all system threads first is a standard running procedure for all Java application skeletons generated with the ASSL framework.

Log Records "Starting System Threads"

```
************************************************************
********** INIT ALL TIERS **********
************************************************************
********** START AS THREADS **********
************************************************************
1)  METRIC 'generatedbyassl.as.aes.ant_ruler.metrics.DISTANCETONEARESTOBJECT': started
2)  EVENT 'generatedbyassl.as.aes.ant_ruler.events.INSTRUMENTLOST': started
3)  EVENT 'generatedbyassl.as.aes.ant_ruler.events.MSGINSTRUMENTBROKENRECEIVED': started
4)  EVENT 'generatedbyassl.as.aes.ant_ruler.events.SPACECRAFTCHECKED': started
5)  EVENT 'generatedbyassl.as.aes.ant_ruler.events.TIMETORECEIVEHEARTBEATMSG': started
6)  EVENT 'generatedbyassl.as.aes.ant_ruler.events.INSTRUMENTOK': started
7)  EVENT 'generatedbyassl.as.aes.ant_ruler.events.MSGHEARTBEATRECEIVED': started
8)  EVENT 'generatedbyassl.as.aes.ant_ruler.events.RECONFIGURATIONDONE': started
9)  EVENT 'generatedbyassl.as.aes.ant_ruler.events.RECONFIGURATIONFAILED': started
10) EVENT 'generatedbyassl.as.aes.ant_ruler.events.COLLISIONHAPPEN': started
11) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INHEARTBEATNOTIFICATION': started
12) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INCOLLISION': started
13) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INTEAMRECONFIGURATION': started
14) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INCHECKINGWORKERINSTRUMENT': started
15) POLICY 'generatedbyassl.as.aes.ant_ruler.aeself_management.SELF_HEALING': started
16) AE 'generatedbyassl.as.aes.ANT_RULER': started
************************************************************
17) METRIC 'generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT': started
18) EVENT 'generatedbyassl.as.aes.ant_worker.events.ISMSGHEARTBEATSENT': started
19) EVENT 'generatedbyassl.as.aes.ant_worker.events.INSTRUMENTCHECKED': started
20) EVENT 'generatedbyassl.as.aes.ant_worker.events.ISMSGINSTRUMENTBROKENSENT': started
21) EVENT 'generatedbyassl.as.aes.ant_worker.events.COLLISIONHAPPEN': started
22) EVENT 'generatedbyassl.as.aes.ant_worker.events.INSTRUMENTBROKEN': started
23) EVENT 'generatedbyassl.as.aes.ant_worker.events.TIMETOSENDHEARTBEATMSG': started
24) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFICATION': started
25) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.ININSTRUMENTBROKEN': started
26) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INCOLLISION': started
27) POLICY 'generatedbyassl.as.aes.ant_worker.aeself_management.SELF_HEALING': started
28) AE 'generatedbyassl.as.aes.ANT_WORKER': started
************************************************************
29) EVENT 'generatedbyassl.as.ants.events.SPACECRAFTLOST': started
30) EVENT 'generatedbyassl.as.ants.events.EARTHNOTIFIED': started
31) FLUENT 'generatedbyassl.as.ants.asself_management.self_healing.INLOSINGSPACECRAFT': started
32) POLICY 'generatedbyassl.as.ants.asself_management.SELF_HEALING': started
33) AS 'generatedbyassl.as.ANTS': started
************************************************************
********** AS STARTED SUCCESSFULLY **********
************************************************************
```

Here, records 1 through to 16 show the ANT_RULER autonomic element startup, records 17 through to 28 show the ANT_WORKER autonomic element startup, and records 29 through to 33 show the last startup steps of the ANTS autonomic system.

After starting up all the threads, the system ran in idle mode for 60 seconds, when the timed event timeToSendHeartbeatMsg occurred. This event is specified in the ANT_Worker to run on a regular time basis every 60 sec (cf. Appendix A). The occurrence of this event activated the self-healing mechanism as shown in the following log records.

Log Records "Self-healing Behavior - Original"

```
************************************************************
********** AS STARTED SUCCESSFULLY **********
************************************************************
34) EVENT 'generatedbyassl.as.aes.ant_worker.events.TIMETOSENDHEARTBEATMSG': has occurred
```

```
35) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been initiated
36) ACTION 'generatedbyassl.as.aes.ant_worker.actions.NOTIFYFORHEARTBEAT': has been performed
37) EVENT 'generatedbyassl.as.aes.ant_worker.events.ISMSGHEARTBEATSENT': has occurred
38) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been terminated

39) EVENT 'generatedbyassl.as.aes.ant_ruler.events.TIMETORECEIVEHEARTBEATMSG': has occurred
40) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been initiated
41) ACTION 'generatedbyassl.as.aes.ant_ruler.actions.CONFIRMHEARTBEAT': has been performed
42) EVENT 'generatedbyassl.as.aes.ant_ruler.events.MSGHEARTBEATRECEIVED': has occurred
43) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been terminated

44) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INCHECKINGWORKERINSTRUMENT': has been initiated
45) ACTION 'generatedbyassl.as.aes.ant_ruler.actions.CHECKWORKERINSTRSTATUS': has been performed
46) EVENT 'generatedbyassl.as.aes.ant_ruler.events.INSTRUMENTOK': has occurred
47) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INCHECKINGWORKERINSTRUMENT': has been terminated

48) EVENT 'generatedbyassl.as.aes.ant_worker.events.TIMETOSENDHEARTBEATMSG': has occurred
49) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been initiated
50) ACTION 'generatedbyassl.as.aes.ant_worker.actions.NOTIFYFORHEARTBEAT': has been performed
51) EVENT 'generatedbyassl.as.aes.ant_worker.events.ISMSGHEARTBEATSENT': has occurred
52) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been terminated

53) EVENT 'generatedbyassl.as.aes.ant_ruler.events.TIMETORECEIVEHEARTBEATMSG': has occurred
54) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been initiated
55) EVENT 'generatedbyassl.as.aes.ant_worker.events.TIMETOSENDHEARTBEATMSG': has occurred

56) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been initiated
57) ACTION 'generatedbyassl.as.aes.ant_ruler.actions.CONFIRMHEARTBEAT': has been performed
58) ACTION 'generatedbyassl.as.aes.ant_worker.actions.NOTIFYFORHEARTBEAT': has been performed
59) EVENT 'generatedbyassl.as.aes.ant_ruler.events.MSGHEARTBEATRECEIVED': has occurred
60) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been terminated

61) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INCHECKINGWORKERINSTRUMENT': has been initiated

62) EVENT 'generatedbyassl.as.aes.ant_worker.events.ISMSGHEARTBEATSENT': has occurred
63) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been terminated

64) ACTION 'generatedbyassl.as.aes.ant_ruler.actions.CHECKWORKERINSTRSTATUS': has been performed
65) EVENT 'generatedbyassl.as.aes.ant_ruler.events.INSTRUMENTOK': has occurred
66) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INCHECKINGWORKERINSTRUMENT': has been terminated
```

As we see from the log records, the self-healing behavior correctly followed the specification model.

Records 34 through to 38 show the initiation and termination of the INHEARTBEATNOTIFICATION fluent. This resulted in the execution of the NOTIFYFORHEARTBEAT action (cf. record 36) that sends a heartbeat message to ANT_Ruler (cf. record 37).

Records 39 through to 43 show how this message is handled by the ANT_Ruler. As it is specified (cf. Appendix A), the timeToReceiveHeartbeatMsg event occurred after 90 seconds running time (cf. record 39). This initiated the INHEARTBEATNOTIFICATION fluent (cf. record 40), which prompted the execution of the CONFIRMHEARTBEAT action (cf. record 41). The latter called a communication function to receive the heart beat message (if any). The message was received and prompted the MSGHEARTBEATRECEIVED event (cf. record 42). The latter terminated the INHEARTBEATNOTIFICATION fluent (cf. record 43).

Records 44 through to 47 show how the INCHECKINGWORKERINSTRUMENT fluent is handled by the system. This fluent is initiated by the MSGHEARTBEATRECEIVED event (cf. Appendix A and record 44). Next the CHECKWORKERINSTRSTATUS action is performed (cf. record 45), which resulted into the INSTRUMENTOK event (cf. record 46). The latter terminated the INCHECKINGWORKERINSTRUMENT fluent (cf. record 47).

Records 48 through to 66 show that the system continued repeating the steps shown in records 34 though to 47. This is because the policy-triggering events are periodic timed events and the system did not encounter any problems performing the executed actions, which could possibly branch the program execution. Note that records 48 through to 66 are not ordered in the same way as records 34 though to 47. This is due to both multithreading nature of the generated application and different periods of the timed events (60 sec and 90 sec). Thus, while the ANT_Ruler was handling the second heartbeat message (cf. record 53), the ANT_Worker was sending the third one (cf. record 55).

This experiment demonstrated that the generated code had correctly followed the specified self-healing policy by reacting to the occurring self-healing events and, thus, providing appropriate self-healing behavior.

Test #2: Simulating Loss of Worker Instrument. In this test, we changed the original ASSL self-healing model for ANTS to simulate the loss of an instrument by the ANT_Worker. Thus, we specified a new inSimulateCollision fluent in the SELF_HEALING policy of the ANT_Worker autonomic element.

In addition, we mapped the inSimulateCollision fluent to a newly specified simulateCollision action. The latter sets the value of the distanceToNearestObject metric to a number violating the metric's threshold class (cf. Figure 3). This causes the collisionHappen event attached to this metric to be prompted and consecutively to initiate the inCollision fluent.

```
1. ACTION simulateCollision {
2.     GUARDS { AESELF_MANAGEMENT.SELF_HEALING.inSimulateCollision }
3.     DOES { SET METRICS.distanceToNearestObject.VALUE = 0.0001 }
4. }

    ....

5. EVENT timeToSimulateCollision { ACTIVATION { PERIOD { 75 SEC } } }
```

Figure 3. Action simulateCollision & Event timeToSimulateCollision

In addition, in order to initiate the inSimulateCollision fluent we specified a timeToSimulateCollision event. The latter is a timed event specified to occur on a regular time basis every 75 seconds (cf. Figure 3).

Another change that we made in the specification model was in the checkANTInstrument action. We modified the action specification to report that the instrument is broken and to trigger the instrumentBroken event. The following log records show the run-time behavior of the new self-healing model for ANTS. Note that we omitted the startup part of the record, which we have already discussed in Test #1.

Log Record "Self-healing with Simulated Loss of Worker Instrument"

```
************************************************************
****************** AS STARTED SUCCESSFULLY *****************
************************************************************
34) EVENT 'generatedbyassl.as.aes.ant_worker.events.TIMETOSENDHEARTBEATMSG': has occurred
35) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been initiated
36) ACTION 'generatedbyassl.as.aes.ant_worker.actions.NOTIFYFORHEARTBEAT': has been performed
37) EVENT 'generatedbyassl.as.aes.ant_worker.events.ISMSGHEARTBEATSENT': has occurred
38) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been terminated

39) EVENT 'generatedbyassl.as.aes.ant_worker.events.TIMETOSIMULATECOLLISION': has occurred
40) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INSIMULATECOLLISION': has been initiated
41) ACTION 'generatedbyassl.as.aes.ant_worker.actions.SIMULATECOLLISION': has been performed
42) EVENT 'generatedbyassl.as.aes.ant_worker.events.COLLISIONHAPPEN': has occurred
43) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INCOLLISION': has been initiated
44) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INSIMULATECOLLISION': has been terminated

    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
45) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

46) ACTION 'generatedbyassl.as.aes.ant_worker.actions.CHECKINSTRUMENT': has been performed
47) EVENT 'generatedbyassl.as.aes.ant_worker.events.INSTRUMENTCHECKED': has occurred
48) ACTION 'generatedbyassl.as.aes.ant_worker.actions.CHECKANTINSTRUMENT': has been performed
49) EVENT 'generatedbyassl.as.aes.ant_worker.events.INSTRUMENTBROKEN': has occurred
50) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.ININSTRUMENTBROKEN': has been initiated
51) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INCOLLISION': has been terminated

    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
52) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

53) ACTION 'generatedbyassl.as.aes.ant_worker.actions.NOTIFYFORBROKENINSTRUMENT': has been performed
54) EVENT 'generatedbyassl.as.aes.ant_worker.events.ISMSGINSTRUMENTBROKENSENT': has occurred
55) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.ININSTRUMENTBROKEN': has been terminated

    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
56) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
57) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
58) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
59) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
60) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
61) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
62) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
63) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
64) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
65) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

66) EVENT 'generatedbyassl.as.aes.ant_ruler.events.TIMETORECEIVEHEARTBEATMSG': has occurred
67) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been initiated
68) ACTION 'generatedbyassl.as.aes.ant_ruler.actions.CONFIRMHEARTBEAT': has been performed

    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
69) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

70) EVENT 'generatedbyassl.as.aes.ant_ruler.events.MSGHEARTBEATRECEIVED': has occurred
71) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INHEARTBEATNOTIFICATION': has been terminated

72) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INCHECKINGWORKERINSTRUMENT': has been initiated
73) ACTION 'generatedbyassl.as.aes.ant_ruler.actions.CHECKWORKERINSTRSTATUS': has been performed
74) EVENT 'generatedbyassl.as.aes.ant_ruler.events.INSTRUMENTOK': has occurred

    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
75) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

76) EVENT 'generatedbyassl.as.aes.ant_ruler.events.MSGINSTRUMENTBROKENRECEIVED': has occurred
77) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INCHECKINGWORKERINSTRUMENT': has been terminated
78) EVENT 'generatedbyassl.as.aes.ant_ruler.events.INSTRUMENTLOST': has occurred

79) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INTEAMRECONFIGURATION': has been initiated
80) ACTION 'generatedbyassl.as.aes.ant_ruler.actions.RECONFIGURETEAM': has been performed
81) EVENT 'generatedbyassl.as.aes.ant_ruler.events.RECONFIGURATIONDONE': has occurred

    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
82) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

83) FLUENT 'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INTEAMRECONFIGURATION': has been terminated

    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
84) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
85) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
```

The following text explains records 34 through to 85.

Records 34 through to 38 are identical with those in Test #1.

Records 39 through to 44 show the initiation and termination of the INSIMULATECOLLISION fluent. This fluent was initiated by the TIMETOSIMULATECOLLISION event (cf. record 39) and prompted the execution of

the SIMULATECOLLISION action (cf. record 41). Next, due to that action changing the distanceToNearestObject metric's value, the COLLISIONHAPPEN event was triggered (cf. record 42). This event terminated the INSIMULATECOLLISION fluent (cf. record 44) and initiated the INCOLLISION fluent (cf. record 43).

Records 45, 52, 56 through to 65, 69, 75, 82, 84, and 85 show that the control loop of the ANT_Worker uncovered a problem with the metric DISTANCETONEARESTOBJECT, and attempted to fix that problem by executing actions. Because, *there was no action set to fix the metric*, the control loop executed a generic action that simply prints a message highlighting that problem (this is the default for all control loops generated with the ASSL framework). In ASSL, control loops monitor and work to fix metrics and service-level-objectives (SLO) of the system. Here, the DISTANCETONEARESTOBJECT metric was discovered as invalid and thus needed to be fixed, because its current value was violating the metric's threshold class. The presence of multiple records of the same type shows that the control loop was constantly trying to fix that problem.

Records 46 through to 51 show the process of checking the ANT_Worker instrument. This resulted in prompting the INSTRUMENTBROKEN event (cf. record 49) and consecutively initiating the ININSTRUMENTBROKEN fluent (cf. record 50). Next, this fluent prompted the action NOTIFYFORBROKENINSTRUMENT (cf. record 53).

Records 66, 67, 68, 70, and 71 show that the ANT_Ruler received the heartbeat message sent by the ANT_Worker (cf. records 34 through to 38).

Records 72, 73, 74, and 77 show the instrument check performed by the ANT_Ruler after receiving the heartbeat message. Note that this check reported the INSTRUMENTOK event (cf. record 74) because the check was based on the heartbeat message sent before the collision.

Record 76 shows that the ANT_Ruler received at that point the message sent by the ANT_Worker and notifying that the worker instrument is broken. This prompted the INSTRUMENTLOST event (cf. record 78) and consecutively initiated the INTEAMRECONFIGURATION fluent (cf. record 79). The latter prompted the execution of the RECONFIGURETEAM action (cf. record 80), which finished with prompting the RECONFIGURATIONDONE event (cf. record 81).

Similar to Test #1, this experiment demonstrated that the generated code had correctly followed the modified self-healing policy by reacting as before to the occurring self-healing events and thus, providing appropriate self-healing behavior.

Test #3: Simulating Worker Loss. In this test, we changed the original ASSL self-healing model for ANTS to simulate loss of an instrument by the ANT_Worker. Thus, we specified a new inSimulateCollision fluent in the SELF_HEALING policy of the ANT_Worker autonomic element.

In this test, we changed the ASSL self-healing model for ANTS from Test #2 to simulate loss of the ANT_Worker. The changes we made in the specification code are as following:

- We set the activation time of the timeToSimulateCollision timed event to 45 seconds, thus simulating a collision before sending the heartbeat message (every 60 seconds).
- We changed the GUARDS clause of the simulateCollision action (cf. Figure 4) to ensure that this action will be performed only once. Thus, we added to that clause the evaluation of the distanceToNearestObject metric, i.e., the action could not perform if that metric is invalid (holds a value that contradicts its threshold class) [2, 7].

```
1. ACTION simulateCollision {
2.     GUARDS { AESELF_MANAGEMENT.SELF_HEALING.inSimulateCollision AND
3.              METRICS.distanceToNearestObject }
4.     DOES {SET METRICS.distanceToNearestObject.VALUE = 0.0001 }
5.     ONERR_TRIGGERS { EVENTS.workerInCollision }
6. }
```

Figure 4. Action simulateCollision (Modified)

Similarly, we changed the GUARDS clause of both the notifyForHeartbeat action and the checkANTInstrument action. This prevented both actions from executing once the distanceToNearestObject metric became *invalid*.

The following log records show the run-time behavior of the modified self-healing model for ANTS. Note that startup part of the records (discussed in Test #1) is omitted here.

Log Record "Self-healing with Simulated Worker Loss"

```
**********************************************************
**************** AS STARTED SUCCESSFULLY ****************
**********************************************************
34) EVENT 'generatedbyassl.as.aes.ant_worker.events.TIMETOSIMULATECOLLISION': has occurred
35) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INSIMULATECOLLISION': has been initiated
36) ACTION 'generatedbyassl.as.aes.ant_worker.actions.SIMULATECOLLISION': has been performed
37) EVENT 'generatedbyassl.as.aes.ant_worker.events.COLLISIONHAPPEN': has occurred
38) FLUENT 'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INCOLLISION': has been initiated
```

```
39) FLUENT
    'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INSIMULATECOLLISI
    ON': has been terminated

    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
40) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

41) ACTION 'generatedbyassl.as.aes.ant_worker.actions.CHECKANTINSTRUMENT': has been
    prevented by GUARDS
42) EVENT 'generatedbyassl.as.aes.ant_worker.events.INSTRUMENTNOTCHECKED': has
    occurred
43) FLUENT
    'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INCOLLISION': has
    been terminated

    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
44) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
45) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
46) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
47) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

48) EVENT 'generatedbyassl.as.aes.ant_worker.events.TIMETOSENDHEARTBEATMSG': has
    occurred
49) FLUENT
    'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFI
    CATION': has been initiated

    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
50) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

51) ACTION 'generatedbyassl.as.aes.ant_worker.actions.NOTIFYFORHEARTBEAT': has been
    prevented by GUARDS
52) EVENT 'generatedbyassl.as.aes.ant_worker.events.HEARTBEATMSGNOTSENT': has
    occurred
53) FLUENT
    'generatedbyassl.as.aes.ant_worker.aeself_management.self_healing.INHEARTBEATNOTIFI
    CATION': has been terminated

    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
54) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
55) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
56) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
57) ACTION 'generatedbyassl.as.ASSLACTION': has been performed
    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
58) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

59) EVENT 'generatedbyassl.as.aes.ant_ruler.events.TIMETORECEIVEHEARTBEATMSG': has
    occurred
60) FLUENT
    'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INHEARTBEATNOTIFIC
    ATION': has been initiated
61) ACTION 'generatedbyassl.as.aes.ant_ruler.actions.CONFIRMHEARTBEAT': has failed to
    fulfill ENSURES post-conditions
62) EVENT 'generatedbyassl.as.ants.events.SPACECRAFTLOST': has occurred
63) FLUENT
    'generatedbyassl.as.aes.ant_ruler.aeself_management.self_healing.INHEARTBEATNOTIFIC
    ATION': has been terminated

64) FLUENT
    'generatedbyassl.as.ants.asself_management.self_healing.INLOSINGSPACECRAFT': has
    been initiated

    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
65) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

66) ACTION 'generatedbyassl.as.ants.actions.NOTIFYEARTH': has been performed

    There is no action set to fix the invalid metric
    generatedbyassl.as.aes.ant_worker.metrics.DISTANCETONEARESTOBJECT'
67) ACTION 'generatedbyassl.as.ASSLACTION': has been performed

68) EVENT 'generatedbyassl.as.ants.events.EARTHNOTIFIED': has occurred
69) FLUENT
    'generatedbyassl.as.ants.asself_management.self_healing.INLOSINGSPACECRAFT': has
    been terminated
```

These log records again show that the generated Java application skeleton provided correct behavior conforming to the specified self-healing policy.

This time, the first event that occurred in the system was the TIMETOSIMULATECOLLISION event (cf. record 34). The latter initiated the INSIMULATECOLLISION fluent, which as in Test #2 simulated a collision in ANT_Worker (cf. records 35 through to 39). This initiated the INCOLLISION fluent (cf. record 38), but his time the CHECKANTINSTRUMENT action did not execute due to its modified GUARDS clause (cf. record 41).

Similarly, the NOTIFYFORHEARTBEAT action did not execute due to its GUARDS clause (cf. record 51) and thus, no heartbeat message was sent (cf. record 52) to the ANT_Ruler.

Records 59 through to 63 show that the ANT_Ruler unable to receive that heartbeat message triggered the SPACECRAFTLOST AS-level event (cf. record 62). The latter initiated the ANTS INLOSINGSPACECRAFT fluent, which notified Earth about the problem (cf. records 64, 66, 68, and 69).

Note that similar to Test #2, the control loop of the ANT_Worker was constantly trying to fix the invalid metric (cf. records 40, 44, 45, etc.).

It is important to mention, that these tests (Test #1, #2, and #3) not only provided strong evidence of valid self-management behavior of the generated code, but also demonstrated the ASSL communication system. Here, messages were successfully sent from one autonomic element (ANT_Worker) and received by another one (ANT_Ruler).

In addition, we have demonstrated the effectiveness of the event-driven self-management policy model, where ASSL events can be associated with messages, metrics, other events, time etc. These events initiate and terminate fluents. The latter prompt the execution of actions.

Moreover, we have demonstrated the effectiveness of the ASSL secure action approach. With conditions specified in the action GUARDS and ENSURES clauses we require certain conditions to be met before and after the action's execution.

7. Conclusion and Future Work

In the most basic of terms, experiments are said to be valid if they do what they are supposed to do. In that context, the experiments and test results described here are valid and they conform to our belief that ASSL framework provides a valid approach for building and validating autonomic systems.

Unfortunately, it is far easier to demonstrate validity of our approach than to demonstrate conclusively its completeness. In part, this is because completeness is at heart a relative rather than an

absolute concept. Therefore, more experiments and results are needed and it is our intention to come up with a more complete ASSL specification model for ANTS emphasizing different autonomic features and to consecutively generate a more complete Java application skeleton for ANTS.

Next, we will complete that generated skeleton to arrive at the first experimental prototype of ANTS. The latter could be extremely useful when undertaking further investigation based on practical results and will help us to test different aspects of autonomic behavior under more simulated conditions.

References

[1] IBM Corporation, "An architectural blueprint for autonomic computing", White paper, 4th ed., 2006.

[2] E. Vassev and J. Paquet, "ASSL - Autonomic System Specification Language", In Proc. of the 31st Annual IEEE/NASA Software Engineering Workshop (SEW-31), IEEE Computer Society Press, March 2007, pp. 300-309.

[3] W. Truszkowski, M. Hinchey, J. Rash, and C. Rouff, "NASA's swarm missions: The challenge of building autonomous software", IT Professional, 6(5), 2004, pp. 47-52.

[4] M. Hinchey, J. Rash, C. Rouff, *Requirements to design to code: Towards a fully formal approach to automatic code generation*, Technical Report TM-2005-212774, NASA Goddard Space Flight Center, Greenbelt.

[5] IBM Tivoli, *Autonomic Computing Policy Language*, Tutorial, IBM Corporation, 2005.

[6] D. Agrawal et al., *Autonomic Computing Expressing Language*, Tutorial, IBM Corporation, 2005.

[7] E. Vassev, *Towards a Framework for Specification and Code Generation of Autonomic Systems*, Ph.D. Thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, November, 2008.

[8] E. Vassev, M. Hinchey, and J. Paquet, "A Self-Scheduling Model for NASA Swarm-Based Exploration Missions using ASSL", *Proceedings of the Fifth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems (EASe'08)*, IEEE Computer Society Press, 2008, pp. 54-64.

[9] E. Vassev, M. Hinchey, and J. Paquet, "Towards an ASSL Specification Model for NASA Swarm-Based Exploration Missions", *Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC 2008) - AC Track*, ACM, 2008, pp.1652-1657.

[10] E. Vassev and M. Hinchey, "ASSL Specification Model for the Image-Processing Behavior in the NASA Voyager Mission", Technical report, Lero—The Irish Software Engineering Research Center, 2009.

Appendix A
ASSL Self-healing Specification for ANTS

```
AS ANTS {
   ASSELF_MANAGEMENT {
      SELF_HEALING {
         FLUENT inLosingSpacecraft {
            INITIATED_BY { EVENTS.spaceCraftLost }
            TERMINATED_BY { EVENTS.earthNotified } }
         MAPPING {  CONDITIONS { inLosingSpacecraft }
            DO_ACTIONS { ACTIONS.notifyEarth } }
      }
   }
   ASARCHITECTURE {
      AELIST {AES.ANT_Worker, AES.ANT_Ruler}
      DIRECT_DEPENDENCIES { DEPENDENCY AES.ANT_Worker { AES.ANT_Ruler }}
      TRANSITIVE_DEPENDENCIES { DEPENDENCY AES.ANT_Ruler  {AES.ANT_Worker }}
      GROUPS {
         GROUP explorerOne {
            MEMBERS { AES.ANT_Worker, AES.ANT_Ruler }
            COUNCIL { AES.ANT_Ruler  }
         }
      }
   }
   ACTIONS {
      ACTION notifyEarth { //notify Earth for the lost spacecraft
         GUARDS { ASSELF_MANAGEMENT.SELF_HEALING.inLosingSpacecraft }
         DOES { CALL ASIP.FUNCTIONS.sendSpacecraftLostMsg }
      }
   }
   EVENTS { // these events are used in the fluents specification
      EVENT spaceCraftLost { }
      EVENT earthNotified { ACTIVATION  { SENT { ASIP.MESSAGES.msgSpacecraftLost } } }
   }
} // AS ANTS

ASIP {
   MESSAGES { MESSAGE msgSpacecraftLost { ... }          }
   CHANNELS { CHANNEL LBW_link { ... } }
   FUNCTIONS { FUNCTION sendSpacecraftLostMsg {... } }
}

//==================== autonomic elements ==================
AES {
   //==================== ANT_Worker ========================
   AE ANT_Worker {
      AESELF_MANAGEMENT {
         SELF_HEALING {
            FLUENT inCollision {
               INITIATED_BY {  EVENTS.collisionHappen  }
               TERMINATED_BY {  EVENTS.instrumentChecked  } }
            FLUENT inInstrumentBroken  {
               INITIATED_BY {  EVENTS.instrumentBroken  }
               TERMINATED_BY {  EVENTS.isMsgInstrumentBrokenSent } }
            FLUENT inHeartbeatNotification  {
               INITIATED_BY {  EVENTS.timeToSendHeartbeatMsg }
               TERMINATED_BY {  EVENTS.isMsgHeartbeatSent }  }
            MAPPING {  CONDITIONS  {  inCollision  }
               DO_ACTIONS  {  ACTIONS.checkANTInstrument  } }
            MAPPING { CONDITIONS  {  inInstrumentBroken  }
               DO_ACTIONS  {  ACTIONS.notifyForBrokenInstrument  } }
            MAPPING  {CONDITIONS  {  inHeartbeatNotification }
               DO_ACTIONS  {  ACTIONS.notifyForHeartbeat  } }
         }
      } // AESELF_MANAGEMENT
      FRIENDS {  AELIST { AES.ANT_Ruler } }
      AEIP {
         MESSAGES {
            FINAL MESSAGE instrumentBrokenMsg  { ....}
            FINAL MESSAGE heartbeatMsg  {.... }
         }
         CHANNELS { CHANNEL HBW_link  {... } }
         FUNCTIONS {
            FUNCTION sendInstrumentBrokenMsg  { .... }
            FUNCTION sendHeartbeatMsg  { .... }
         }
         MANAGED_ELEMENTS {
            MANAGED_ELEMENT worker {
               INTERFACE_FUNCTION getDistanceToNearestObject { RETURNS { DECIMAL }
               }
            }
      } // AEIP
      ACTIONS {
         ACTION IMPL checkInstrument {
            RETURNS {  BOOLEAN  }
            TRIGGERS {  EVENTS.instrumentChecked }
         }
         ACTION checkANTInstrument {
            GUARDS { AESELF_MANAGEMENT.SELF_HEALING.inCollision }
            VARS { BOOLEAN canOperate }
            DOES { canOperate = CALL ACTIONS.checkInstrument }
            TRIGGERS { IF (not canOperate) THEN EVENTS.instrumentBroken END }
         }
         ACTION notifyForBrokenInstrument {
            GUARDS { AESELF_MANAGEMENT.SELF_HEALING.inInstrumentBroken }
            DOES { CALL AEIP.FUNCTIONS.sendInstrumentBrokenMsg }
         }
         ACTION notifyForHeartbeat {
            GUARDS { AESELF_MANAGEMENT.SELF_HEALING.inHeartbeatNotification }
            DOES { CALL AEIP.FUNCTIONS.sendHeartbeatMsg }
         }
      } // ACTIONS
      EVENTS { // these events are used in the fluents? specification
         EVENT collisionHappen {
            GUARDS { not METRICS.distanceToNearestObject }
            ACTIVATION  { CHANGED  { METRICS.distanceToNearestObject } } }
         EVENT isMsgInstrumentBrokenSent {
            ACTIVATION  { SENT { AEIP.MESSAGES.instrumentBrokenMsg } } }
         EVENT instrumentBroken { }
         EVENT instrumentChecked { }
         EVENT timeToSendHeartbeatMsg { ACTIVATION  {  PERIOD  {  1 min } } }
         EVENT isMsgHeartbeatSent {
            ACTIVATION  { SENT { AEIP.MESSAGES.heartbeatMsg } } }
      }
      METRICS {
         METRIC distanceToNearestObject  {  METRIC_TYPE  {  RESOURCE  }
            METRIC_SOURCE {
               MEIP.MANAGED_ELEMENTS.worker.getDistanceToNearestObject }
            MEASURE_UNIT  { "KM" }
            THRESHOLD_CLASS  {  DECIMAL  [0.001 ~ )  }
         }
      }
   } // AE ANT_Worker

   //==================== ANT_Ruler ========================
   AE ANT_Ruler {
      AESELF_MANAGEMENT {
         SELF_HEALING {
            FLUENT inCollision {
               INITIATED_BY { EVENTS.collisionHappen }
               TERMINATED_BY { EVENTS.spacecraftChecked,AS.EVENTS.spaceCraftLost}
            }
            FLUENT inHeartbeatNotification {
               INITIATED_BY { EVENTS.timeToReceiveHeartbeatMsg }
            TERMINATED_BY {EVENTS.msgHeartbeatReceived,AS.EVENTS.spaceCraftLost}
            }
            FLUENT inCheckingWorkerInstrument {
              INITIATED_BY { EVENTS.msgHeartbeatReceived }
              TERMINATED_BY { EVENTS.instrumentOK, EVENTS.instrumentLost }
            }
            FLUENT inTeamReconfiguration {
               INITIATED_BY { EVENTS.instrumentLost }
            TERMINATED_BY {EVENTS.reconfigurationDone,EVENTS.reconfigurationFailed}
            }
            MAPPING {  CONDITIONS  { inCollision }
               DO_ACTIONS  { ACTIONS.checkSpacecraft }  }
            MAPPING  {     CONDITIONS  { inHeartbeatNotification }
               DO_ACTIONS { ACTIONS.confirmHeartbeat }  }
            MAPPING  {     CONDITIONS  { inCheckingWorkerInstrument }
               DO_ACTIONS  { ACTIONS.checkWorkerInstrStatus }  }
            MAPPING  {  CONDITIONS  { inTeamReconfiguration }
               DO_ACTIONS  { ACTIONS.reconfigureTeam }  }
         }
      }
      AEIP {
         FUNCTIONS {
            FUNCTION receiveHeartbeatMsg  { ....}
            FUNCTION receivedInstrumentBrokenMsg  {....}
         }
         MANAGED_ELEMENTS {
            MANAGED_ELEMENT ruler {
               INTERFACE_FUNCTION getDistanceToNearestObject { RETURNS { DECIMAL}}
            }
         }
      }
      ACTIONS {
         ACTION IMPL checkSpacecraft {
            TRIGGERS { EVENTS.spacecraftChecked }
            ONERR_TRIGGERS { AS.EVENTS.spaceCraftLost }
         }
         ACTION confirmHeartbeat {
            GUARDS { AESELF_MANAGEMENT.SELF_HEALING.inHeartbeatNotification }
            DOES { CALL AEIP.FUNCTIONS.receiveHeartbeatMsg }
            ONERR_TRIGGERS { AS.EVENTS.spaceCraftLost }
         }
         ACTION checkWorkerInstrStatus {
            GUARDS {ESELF_MANAGEMENT.SELF_HEALING.inCheckingWorkerInstrument}
            DOES { CALL AEIP.FUNCTIONS.receivedInstrumentBrokenMsg }
            TRIGGERS {  IF EVENTS.msgInstrumentBrokenReceived THEN
               EVENTS.instrumentLost END  ELSE EVENTS.instrumentOK END }
         }
         ACTION IMPL reconfigureTeam {
            GUARDS { AESELF_MANAGEMENT.SELF_HEALING.inTeamReconfiguration }
            TRIGGERS { EVENTS.reconfigurationDone }
            ONERR_TRIGGERS { EVENTS.reconfigurationFailed }
         }
      }
      EVENTS { // these events are used in the fluents? specification
         EVENT collisionHappen {   GUARDS { not METRICS.distanceToNearestObject }
           ACTIVATION  { CHANGED  { METRICS.distanceToNearestObject } }
         }
         EVENT spacecraftChecked { }
         EVENT timeToReceiveHeartbeatMsg { ACTIVATION  {  PERIOD  {  90 sec } } }
         EVENT msgHeartbeatReceived {
           ACTIVATION  { RECEIVED { AES.ANT_Worker.AEIP.MESSAGES.heartbeatMsg } } }
         EVENT msgInstrumentBrokenReceived {
ACTIVATION  { RECEIVED { ES.ANT_worker.AEIP.MESSAGES.instrumentBrokenMsg }}}
         EVENT instrumentOK {}
         EVENT instrumentLost {
           ACTIVATION  { OCCURRED { EVENTS.msgInstrumentBrokenReceived } } }
         EVENT reconfigurationDone {}
         EVENT reconfigurationFailed {}
      } // EVENTS
      METRICS {  METRIC distanceToNearestObject  {.... } }
   } // AE ANT_Ruler
}
```